EvoEvo Deliverable 3.4

Paulien Hogeweg, Guillaume Beslon, Susan Stepney

EvoEvo Deliverable 3.4

Evolution of open-endedness; Mechanisms and consequences

Due date:	M30
Person in charge:	Paulien Hogeweg
Partner in charge:	UU
Workpackage:	WP3 (In silico experimental study of EvoEvo)
Deliverable description:	Evolution of open-endedness; Mechanisms and consequences: A report describing how open-endedness evolves in the model and its consequences on evolutionary dynamics..

Revisions:

Revision no.	Revision description	Date	Person in charge
1.0	First version of the report	07/10/16	G. Beslon (INRIA)
1.1	New introduction; new sections 8.1 and 8.2	21/10/16	G. Beslon (INRIA)
1.2	New section 8.3	25/10/16	P. Hogeweg (UU)
1.3	Reorganization of the introduction	28/10/16	S. Stepney (UoY)
1.4	Reorganization of the conclusion	28/10/16	G. Beslon (INRIA)
1.5	Corrections and validation	28/10/16	G. Beslon (INRIA)

Table of Contents

ABSTRACT 4

1. INTRODUCTION 4

1.1. BACKGROUND 4
1.2. MOTIVATION 4
1.3. OUTLINE 5

2. DEFINITIONS OF OPEN-ENDEDNESS 6

2.1. A LOOK AT THE LITERATURE 7

3. A META-MODEL FOR OPEN-ENDEDNESS 9

3.1. MODELS AND META-MODELS 9
3.1.1. SCIENTIFIC MODELS 9
3.1.2. ENGINEERING MODELS 10
3.1.3. COMPUTATIONAL MODELS 11
3.1.4. META-MODELS 11
3.2. AN ENTITY-BASED MULTI-LEVEL META-MODEL 12
3.2.1. DOMAIN OF INTEREST 12
3.2.2. ENTITIES 13
3.2.3. ENVIRONMENT 13
3.2.4. SYSTEM 13
3.2.5. INTERACTIONS 14
3.2.6. LEVELS 14
3.2.7. HIERARCHIES AND EMERGENCE IN THE NATURAL WORLD AND IN HUMAN SYSTEMS 16
3.2.8. INFORMATION 19
3.2.9. TIMESCALES 20

4. OPEN-ENDEDNESS AND NOVELTY 20

4.1. TYPES OF NOVELTY 20
4.1.1. TYPE-0 NOVELTY: VARIATION 21
4.1.2. TYPE-1 NOVELTY: INNOVATION 21
4.1.3. TYPE-2 NOVELTY: EMERGENCE 22
4.1.4. CLASSIFICATION VERSUS MEASURES 23
4.2. EVENTS AND CLASSES OF EVENTS 23
4.3. OPEN-ENDED EVENTS AND OPEN-ENDED SYSTEMS 24

5. LIMITS TO OPEN-ENDEDNESS 25

5.1. UPWARD LIMITS 25
5.2. DOWNWARD LIMITS 26
5.3. EFFECTIVELY OPEN-ENDED 26

6. SIMULATION OF OPEN-ENDEDNESS 27

6.1. GENERATIVE LAYER 28
6.2. SHORTCUTS 28
6.2.1. INDIVIDUALITY 29
6.2.2. REPLICATION 29
6.2.3. FITNESS 29
6.3. A GENERAL ARCHITECTURE FOR OPEN-ENDED SIMULATIONS 30
6.4. CHANGING THE MODEL 31
6.5. CAPTURING THE NOVELTIES 32
6.6. DESIGN PRINCIPLES 33

7. ILLUSTRATIVE EXAMPLES 35

7.1. GAME OF LIFE 35
7.2. EXPERIMENTAL EVOLUTION 36
7.3. GENETIC ALGORITHMS 37
7.4. GENETIC PROGRAMMING 39
7.5. COREWORLDS/AUTOMATA CHEMISTRIES 39

8. INSIGHTS FROM THE EVOEVO MODELS 42

8.1. INTRODUCTION 42
8.2. EVO^{2}SIM 42
8.3. EMERGENCE OF SPATIO-TEMPORAL STRUCTURES 47

9. CONCLUSIONS 49

9.1. SUMMARY 49
9.2. DISCUSSION 50

10. REFERENCES 50

Abstract

In EvoEvo we are concerned, among other aspects of evolution, with the **open-endedness** of the evolutionary process itself. How do biological organisms evolve in an open-ended manner? How can our biological models of such systems capture this open-endedness? How can we incorporate this open-endedness into bio-inspired algorithms?

Before we can start to address such questions, we need to clarify the very concept under investigation: “open-endedness”. That is the main purpose of this deliverable.

The open-endedness of a system is often defined as a continual production of **novelty**, but this definition simply begs the question. What is “novelty”? Here we pin down this concept more fully by defining several types of novelty that a system may exhibit, classified as **variation**, **innovation**, and **emergence**. We then provide a **meta-model** for including **levels** of structure in a system’s model. From there, we define an architecture suitable for building **simulations** of open-ended novelty-generating systems, and discuss how previously proposed systems fit into this framework. We conclude by demonstrating how this definition can be used to analyze two of the models developed in the EvoEvo project.

1. Introduction

1.1. Background

Open-endedness, roughly defined as “the ability to continuously produce novelty and/or complexity”, is considered a ubiquitous feature of biological, technosocial, cultural systems, and many other complex systems that develop in a self-organized manner. There is ample literature arguing that both natural and artificial systems share this feature in various domains and that examples of open-ended systems are provided by biological evolution, human languages, legal systems, economic and financial systems, music and art, science and mathematics, to name only a few.

In the field of artificial life (ALife), open-ended evolution and general open-endedness are crucial concepts, whose quest and fulfillment constitute one of its “millennium prize problems” (Bedau et al., 2000). What is surprising, however, is that open-endedness has not found a sufficiently clear-cut and stringent definition but remains ill-defined, despite more than 20 years of efforts to study its realization.

1.2. Motivation

Open-ended evolution seems to be one of the fundamental characteristics of life, perhaps even a defining characteristic, motivating the search for a deeper understanding of the phenomenon. But in recent years the problem has taken on an added urgency, because the open-endedness of human socio-economico-technological systems has begun to have profound consequences on earth’s ecosphere as a whole: we are now living in the Anthropocene.

We contend that open-endedness is essentially a generic property, one that can be best understood if studied not just in one domain, biology, but in all the types of systems in which it operates, including artificial ones. Such a generic formal framework would also be an important step toward a unified scientific treatment of natural-human systems.

Informally, open-endedness refers to the ability of a system to continue producing "interesting" novelty without exhaustion[1]. In this paper we aim to advance the study of open-endedness from both a scientific and an engineering perspective. Classically, the former tries to understand and analyze phenomena, while the latter tries to implement, modify, and manage them. However, in the context of computational sciences and particularly ALife, one may be interested in implementing the phenomenon in order to understand it. That is why we here specifically discuss the conditions for a suitable implementation of open-endedness.

In the context of the EvoEvo project, there are several and diverse reasons to study open-endedness. These include the following:

- We want to explain evolution in vivo: What are the reasons for the actual path of evolution observed on Earth? In particular, do the "Major Evolutionary Transitions" (Maynard Smith and Szathmary, 1995) that occurred in the actual evolutionary history follow some specific rules that would be those of open-endedness (if any)? Or is it contradictory even to speak of "rules" of open-endedness, as "rules" tacitly exclude or constrain some otherwise "open" possibilities?
- We want to elucidate fundamental and methodological aspects of ALife: Can we engineer or manage "major" transitions (Maynard Smith and Szathmary, 1995) in technical, social, or biological systems by building models able to represent them – but without hard-coding these transitions explicitly?
- We want to realize the property of open-endedness in algorithms: Can the "universality" of physical computers support the generation of artificial open-endedness? Do results from the theory of computation and complexity in computer science apply to open-endedness?
- We want to feed back into bio-inspired engineering the concepts of biological complexity: Is it possible to design and implement creative machines?

Hence, there is a need for a sufficiently precise definition of open-endedness to ensure that those interested in this topic can agree on what they observe, or the nature of the problem(s) they are actually trying to solve. Further, requirements need to be established for open-endedness to ultimately become a measurable and quantifiable feature of a system. Artificially designed systems will help in this process of refinement as they allow for well-prepared complex systems to be observed and manipulated in arbitrary ways.

1.3. Outline

The structure of this deliverable is as follows. Before presenting our own definition of open-endedness, we need to install the concept of open-endedness on better foundations, hence

[1] A review of the definitions found in the literature is provided in section 2.1

start with a review of the literature on the topic (section 2). Then, as our elements of definition are necessarily relative to some model, we introduce a methodological discussion of models and meta-models, and describe the main features of our own model: entities, systems, levels, information, and timescales (section 3). This establishes a framework in which open-endedness can be defined using three classes of novelty (section 4).

This definition also exposes certain limits to open-endedness from various perspectives, including issues of computational simulation (section 5). Given these limits, we augment our model of open-ended systems with two engineering components in order to provide an architecture for effective simulations of open-endedness (section 6). We then discuss a number of examples from the literature in terms of our framework and associated architecture (section 7). We finally illustrate our overall arguments on the basis of the evolutionary models developed in the context of the EvoEvo project and analyze the results of this model in terms of open-endedness (section 8).

Sections 2–7 have been published as (Banzhaf *et al.,* 2016b), having been developed during an EvoEvo workshop held in the summer of 2014, and subsequent project meetings. Section 8.2 is being prepared for publication as (Rocabert et al., 2016a) and section 8.3 is being prepared for publication as (Hickinbotham et al*.,* 2016). Elements of the argument proposed here have also been published in the context of the first Open-Endedness workshop organized by Mark Bedau, Tim Taylor and Alastair Channon as a satellite event of ECAL'2015 (Taylor et al., 2016) and of the second Open-Endedness workshop organized as a satellite event of ALIFE'2016 (Banzhaf et al., 2016a). Other elements have been presented at the 2nd EvoEvo Workshop (Stepney and Beslon, 2016)

2. Definitions of Open-Endedness

"Open-ended" may refer to different properties of a system's process. At least two major categories of open-endedness, based on two meanings of "end", can be distinguished: (1) processes that do not stop, and (2) processes that do not have a specific objective. Biological evolution clearly fulfills both criteria, but they need not always be correlated. Absence of time limit within a small set of possible outcomes (or a single outcome) is the hallmark of attractor dynamics, by which a system settles into one stable or stationary pattern, such as a limit cycle or a chaotic trajectory: while it is still running on the short timescale, it has nevertheless reached an end state. Conversely, absence of goal within a bounded duration would be the case of a stochastic system such as a roll of the dice or a bagatelle (a board game in which a ball bounces on pins): these are short-lived but can basically "land anywhere".

Combining these two categories of open-endedness (as in biological evolution) creates a third category, which is likely to be the most appropriate one from our point of view: (3) processes that do not stop and have no specific objective. This position emphasizes the transient part of dynamical systems, which in many cases can be extremely protracted to the point of never reaching convergence. Much of what goes on in living systems could be characterized as an "attempt" to remain in that transient part.

2.1. A Look at the Literature

There is substantial scientific literature using the expressions “open-endedness” (OE) or “open-ended evolution” (i.e. OE in the context of biological evolution) from which some general features are visible.

Despite the fact that the concept strongly involves reference to biological systems and evolution in the biosphere (Bedau, 1999), literature on OE mainly originates from the fields of ALife and evolutionary computation. This contrasts with the idea conveyed by many papers that life is obviously open-ended (Bedau, 1996; Bedau et al., 1998; Huneman, 2012; Maley, 1999; Nehaniv et al., 2006; Sipper et al., 1997; Sipper et al., 1998; Skusa and Bedau, 2002; Stepney and Hoverd, 2011). Moreover, except for a few seminal contributions (Barricelli, 1962; Bedau, 1991; Harvey, 1992; Ray, 1992), most of this literature is from the last 20 years. However, many classical questions in evolutionary biology can be considered close to the question of OE. Notable examples from the biological literature are (Maynard-Smith, 1988; Rensch, 1959; Waddington, 2008), which are discussions of OE in all but name.

Although OE is a central concept in many scientific articles and communications, very few authors risk a definition that goes beyond the couple of sentences of general papers in which OE is also not defined precisely, for example, (Bedau et al., 2000). Some have even referred to “truly open-ended evolution” without clarifying what it means.

When we take a closer look at the various definitions of OE proposed in the literature, four different categories emerge: OE can be defined as: (i) “perpetual production of novelty”; (ii) “unbounded evolution”; (iii) “continual production of complexity”; or (iv) “the essence of life[2]”. We discuss these categories one by one:

(i) **OE as perpetual production of novelty**. This is the most classical definition of OE (Baptista and Costa, 2013; Bianco and Nolfi, 2004; Lehman and Stanley (2011); Rasmussen et al., 2004; Ruiz-Mirazo et al., 2008; Stepney and Hoverd, 2011). It mainly follows Taylor’s work (Taylor, 1999), although some earlier references can also be found (Kaneko, 1994). Taylor defines an open-ended evolutionary system as “a system in which components continue to evolve new forms continuously, rather than grinding to a halt when some sort of ‘optimal’ or stable position is reached”. Interestingly, he adds that “open-ended evolution does not necessarily imply any sort of evolutionary progress”, which clearly distinguishes this definition from the ones based on evolution of complexity (see third item below).

(ii) **OE as unbounded evolution.** This definition of OE is due to the seminal work of Bedau (Bedau, 1991; Bedau and Packard, 1992; Bedau et al., 1998), illustrated in an ecological model of L-system plants (Fernandez et al., 2012) where evolution is deemed “without a definite maximum fitness, hence no optimal goal or solution to reach”, or in abstract heterogenous cellular automata (Medernach et al., 2013),

[2] Note, however, that the last definition may not be exclusive. In particular, one can define open-ended evolution as an unbounded evolutionary process and simultaneously consider that open-ended evolution is a definition of life.

where it is qualified as “long-term”. It is close to (i), but complemented by statistical measures to characterize long-term evolutionary dynamics and distinguishes between three classes of evolutionary systems, including a test for OE. These measures have been refined by Channon (2003), and a fourth class has been added by Skusa and Bedau (2002). A few models have successfully passed the given test (Channon, 2001; Maley, 1999). However, their long-term dynamics are far from the level of diversity and complexity observed in natural systems, thus raising questions about the test itself and about the equivalence of unboundedness and open-endedness of evolution (Maley, 1999).

(iii) **OE as continual production of complexity.** In the previous definitions of OE, there is no explicit qualitative characterization of the kind of novelty produced by evolution. In this third definition, only novelties that increase the complexity of the evolving entities are considered. This definition of OE as a continual production of complexity is popular (Bentley, 2003; Fernando et al., 2011; Heylighen, 2012; Hutton, 2002; Lehman and Stanley, 2012; Ruiz-Mirazo and Moreno, 2012; Schulman et al., 2012). However, this definition, which implicitly requires the existence of an “arrow of complexity” in biological evolution, is rejected by some authors, who state that open-ended evolution does not imply an increase of complexity but, simply, creates the possibility for it (Markovitch et al., 2012; Ruiz-Mirazo et al., 2004; Taylor, 1999). Other authors suggest that the issues of OE (considered as continual production of novelty) and of complexity increase are linked because simple agents in artificial life have less possibility of producing novelty (Soros and Stanley, 2014; Standish, 2003), though this is clearly not the case in biological systems, given the history of bacteria.

(iv) **OE as the essence of life.** Although not widely accepted in the literature, the idea that open-ended evolution could be considered as a definition of life is proposed in many papers. Ray writes that he “would consider a system to be living if it is self-replicating, and capable of open-ended evolution” (Ray, 1992), an idea that was further discussed by Bedau (1996). For Ruiz-Mirazo et al. (2004), a living system is considered to be an autonomous system with the property of OE brought about by a process of evolution.

All four definitions are similar to each other. In many contributions, the difference between ‘novelty’ and ‘complexity’ is not clear at all. Moreover, they all originate from considerations on biological evolution per se, whereas the phenomenon of OE is widespread in the universe, as we stated earlier. Most importantly, these definitions generally suffer from a major drawback: they are based on terms whose definition is itself challenging (‘novelty’, ‘complexity’, ‘unbounded’, ‘life’), as attested by qualifications added to the meaning of these terms. For example, Bianco and Nolfi (2004) use the expression ‘major novelty’, while Rasmussen et al. (2004) speak of ‘adaptive novelty’. Similarly, Taylor (1999) suggests that continuous production of novelty means that “an indefinite variety of phenotypes are attainable through the evolutionary process, rather than continuous change being achieved by, e.g. cycling through a finite set of possible forms”. Thus, not all forms of novelty would lead to OE.

We therefore arrive at two contrasting definitions. On the one hand, there is the idea of continual (unbounded) creation of novelty, which appears to be necessary but not sufficient. On the other hand, there is the notion of continual (unbounded) creation of complexity, which appears to be sufficient, but not necessary for OE.

One of the motivations of this paper is to offer a new definition that avoids these pitfalls. To this end, we propose to keep the first idea above, that of "continual generation of novelty", but reposition it in the well-defined context of models and modeling. We argue that the concept of a model is fundamental to the definition and analysis of OE and open-ended systems, because the very notion of novelty cannot be understood without the requirement for a model of the observed (and changing) system. The modeling aspect is sufficiently general to capture non-biological systems as well, and allows us to see them from the perspective of OE.

In the next section, we discuss this point further by introducing the notion of 'models' and 'meta-models'. Then, we give a few basic definitions of the elements that constitute the model that we later used to define OE.

3. A Meta-Model for Open-Endedness

"All models are wrong, but some are useful"

– George E. P. Box, 1987

3.1. Models and Meta-Models

3.1.1. Scientific models

Scientific models are descriptive models of part of the existing world. Physical reality comprises material and energy with structure and dynamics, exhibiting patterns in structure and behavior. We perceive some of these patterns (and even non-existent pseudo-patterns), and build models of the world.

The models are abstractions; we overfit and underfit, abstract and omit, err, confabulate and imagine. Our models range from implicit mental pattern recognition and expectations, through prose, informal cartoons, sketches and diagrams, to fully formal computational and mathematical models. But they are all models, and reality can be different (richer, poorer, other). In particular, our models tend to be "crisper" than reality, identifying classes and categories where there are actually spectra, and having difficulty with "borderline" structures that fall between or on the boundaries of our categories (for example, viruses are problematic if we are using a model that insists life is a binary property).

In scientific models, if reality and our model disagree unacceptably for our purposes, it is the model that is wrong. The model needs to be corrected. We can use our models to classify, understand, explain, and predict. Things can happen in the world that are outside the model. For example, a model of traffic flow might not include the current phase of the moon – unless it explicitly considers the effect of moon luminosity on traffic. At other times, things can

happen that are in the scope of the model, but the model does not capture them sufficiently well (for example, borderline entities). And things can happen that start in the model, but move outside the model (for example, the evolution of a new species of entity), requiring a modification of the model to capture the new features. Hence, our scientific understanding of the system is model-dependent (for example, if we were already using a model of the system as it exists after a speciation, then the speciation event would not move outside the model).

There are many model-types used to categorize the roles that models play in scientific practice, for example, phenomenological models, computational models, scale models, analogue models. For a philosophical overview of categorizing models in science, see [41, 113]. The rise of computer simulations has significantly shaped our understanding of models and their relation to theory and experiments. These issues are addressed in (Frigg and Hartmann, 2012; Winsberg, 2010) and (Winsberg, 2013, ch.4). For a discussion of developing false models as a means to making scientific progress, see (Wimsatt, 1987).

3.1.2. Engineering models

Engineering models are prescriptive or normative models of a system to be built in the world, for example, a bridge.

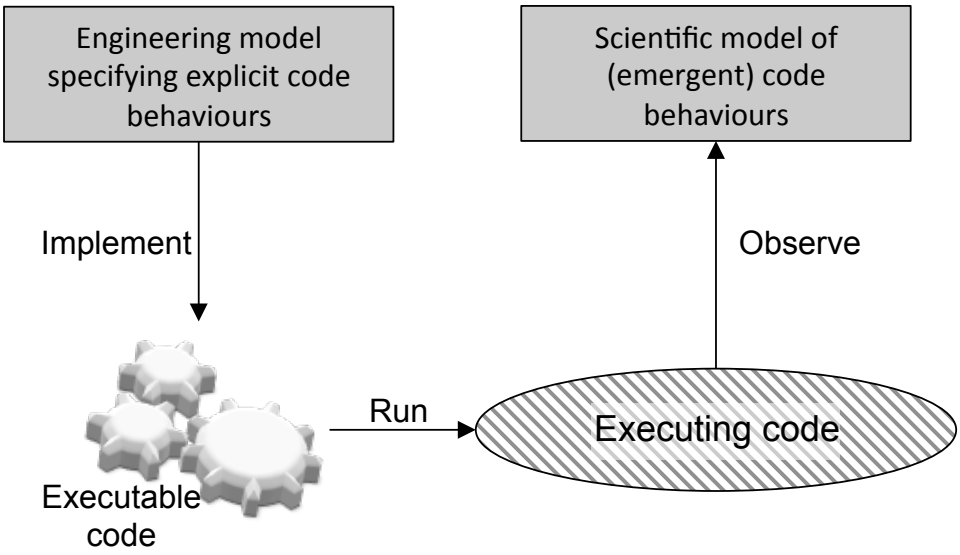

Figure 1: Two kinds of computational model. An engineering model is used to specify the software to be implemented. A scientific model is derived from observations of the execution of the software. These models can have different structures, even about the same software.

When building a system, we construct a model of what the system is meant to do. An engineering model describes what we are trying to bring into existence, in contrast to a post hoc scientific model of existing reality. It is still a model, however, in that it may not capture exactly what the system does. We are here assuming for simplicity that the underlying scientific model of the relevant domain is well understood and correctly applied. More complete accounts are available, for example (Horsman et al., 2014).

In engineering, if reality (the engineered system) and the model of the system disagree, it is reality that is wrong, for example, if the bridge collapses. Reality needs to be "debugged" to conform more closely to the model.

3.1.3. Computational models

Software is a special case of an engineered system, with the model being the specification and design of the software, and the engineered system being the executing code. For example, in an object-oriented system, the model could be a collection of UML diagrams. The software is the executing code that conforms to this model (coded classes corresponding to boxes in the class diagram, with behaviors following the sequence diagrams). If no engineering model pre-exists, if the model is implicit and only the code is written, then a post hoc scientific-style model can be inferred. Such a model will be much messier than a designed engineering model.

Computational science and ALife can involve many different intertwined models, both engineering and scientific. In computational science, software can be used as an executable model of a real object (biological, sociological, technological, etc.). The model of interest is the scientific model built from observing the execution of the code (analogous to the scientific models built from observing real world behaviors). Additionally, the software will generally have been designed using engineering models (for example, UML diagrams). See figure 1.

So computational models can have different content depending on the specific objectives: scientific or engineering. For example, if the objective is to engineer an open-ended software system, the property of OE will be part of the engineering model of the code. Alternatively, if the objective is to model the observed behavior of the simulation of a real world open-ended system (in the context of computational science), then OE may be present in the scientific model, derived from observing emergent behavior in the execution of the code, but it need not be present in the prior engineering model of the code. Indeed, it should not be present in the engineering model, if the objective is to determine if certain low-level behaviors can exhibit certain emergent properties.

3.1.4. Meta-models

As discussed above, the purpose of a model is to provide an abstract language for the relevant domain concepts. In the case of a computational model, it defines the concepts to be implemented in code. The purpose of a meta-model is to provide the analogous language to define such models, to provide the concepts that can be used to build the model. For example, in an object-oriented system, the meta-model would contain the concepts of 'class', 'object', 'method', 'association', and so on. Similarly, a diagram describing the Krebs cycle of an organism is a model whose meta-model contains the concepts of 'reactions', 'metabolites', or 'enzymes'. See (Hoverd and Stepney, 2011) for an example of a meta-model capturing concepts of 'energy', 'environment', and 'organism', and its instantiation in an agent-based model. Roughly speaking, the meta-model is the key to the model in the way the key to a map lists all the graphical concepts used to draw the map. In many modeling contexts the relevant meta-models may be implicit.

A variety of models may conform to the same meta-model. For example, the meta-model used by the model of the Krebs cycle can be used by models of other metabolic pathways. On the other hand, the same system can be captured by a variety of models conforming to different meta-models. For example, a metabolic pathway can be modeled by a set of Ordinary Differential Equations (ODEs), with a meta-model including concepts such as 'concentration' and 'enzymatic rate' (Andrews et al., 2011). Metabolic pathways can also be modeled using agent-based formalisms (Amar et al., 2008), with a meta-model including concepts such as 'agent' and 'rule'. The choice of a meta-model directly determines the limits of the models that could conform to it. In the example, the meta-model allowing ODE descriptions and resulting ODE models cannot account for spatial behavior of the metabolic pathway or for its stochastic behavior. A meta-model allowing agent-based formalisms admits models which can naturally include both behaviors.

In essence, a meta-model is a model of a model (Kleppe et al., 2003, ch.8). Since a model may itself be a meta-model, there is no need for a separate concept of a meta-meta-model, although there may be examples of such. For example, UML is the meta-model of the meta-model presented in Figure 3. Similarly, one can view this paper as an (informal) meta-model of open-endedness.

3.2. An Entity-Based Multi-Level Meta-Model

OE is a process by which continual novelty is produced in the course of a dynamic process. In this paper we propose a scientific (descriptive) meta-model with which we illustrate our definition of OE. This particular entity-based multi-level meta-model should cover evolution in biological life, change in socioeconomic systems, and processes in ALife. Although ALife concerns artificial systems, our meta-model is not a prescriptive (normative) engineering model, but a model for describing open-ended systems, natural or artificial. Other meta-models of OE are probably possible, for example, ones supporting mathematical models formulated in terms of ODEs or PDEs. Our hope is that such a definition of OE in terms of models and meta-models will help the design of normative engineering models for implementing ALife, and provide tools to describe and understand the long-term dynamic of Open-Ended systems.

In our models we are working with systems (biological, social, artificial) of entities organized into multiple levels. In the following subsections, we introduce basic definitions with respect to some (meta-)model. In a given situation, the chosen model must be a sufficiently good representation of reality to be useful.

3.2.1. Domain of interest

The purpose of our meta-model is to provide a language able to model any open-ended system. As explained below, this system can itself be composed of (sub-)systems. We use the term "domain of interest" to describe the open-ended system under investigation, and in order to avoid confusion between the system to be modeled and the systems comprising the model. Note that the concept of domain of interest is not part of our meta-model (Figure 3), but rather a limit between what we want to study and what remains out of scrutiny. Since we

do not wish to model the entire universe, the domain of interest encompasses the part of the world that we are modeling.

3.2.2. Entities

An **entity** is an identifiable integrated whole within the model: a "thing" with structure (organization) and behavior (activity, processing). Entities can **interact** with each other, and with their **environment**. We explicitly sidestep any discussion of the non-trivial issue of identification or demarcation of entities (Buss, 1987).

Note that entities may themselves contain entities. This property naturally leads to the concept of **levels**:

- A **level-0 entity** is an **atomic entity**, with no modeled internal structure.
- A **level-N entity** (where N > 0) is a **system entity**, having lower-level entity components.

The concept of levels is refined further in section 3.2.6.

3.2.3. Environment

An **environment** is a model of part of the domain of interest not modeled as explicit entities, including space, fields, flows, and so on. What is part of an environment and what is an entity is a modeling decision, and depends on what entities are chosen to be on the lowest level of our systemic hierarchy (section 3.2.6).

- **A local environment** is the part of a system that is not modeled explicitly as entities.
- **An external environment** is that part of the domain of interest which is external to the system but exerts an influence on it. For example, it might impinge on the system via inputs (flows, forces, signals) and be in turn influenced by the system via outputs. The external environment may host other entities external to the system.

3.2.4. System

A **system** is a **local environment** plus a population of possibly differentiated and interacting entities, forming some identifiable whole, and separated from an external environment.

A system may be an **aggregate system**, comprising a collection or population of entities in a local environment, but not considered to form an entity in its own right. For example, a simple population of organisms is usually an aggregate system, and not an entity in its own right[3].

Alternatively, a system may be a **system entity**, an entity at a higher **level** than its component self-organized entities, constraining them and their local environment via downward causation.

An **atomic entity** is not a system, as it has no internal (modeled) structure.

[3] Though proponents of group selection, who claim that natural selection can act on populations and not just on individuals (Wilson, 1997), would disagree.

The specific entities comprising the system's components may change over time, for example, by recycling material. Also, specific entities may belong to more than one system, either separated in time as entities flowing from one system to another, or simultaneously. For example, subsystems may be composed into a larger system by overlapping/intersecting some of their entities. A system needs to have a boundary in some space; typical such spaces include physical space, time, speed, or behavior.

3.2.5. Interactions

Entities **interact** with each other, and with their environment, potentially forming and dissolving systems.

3.2.6. Levels

We have defined a system entity to be an entity at a higher level than its component entities. Entities can be recursively classified into levels with corresponding interactions (see Figure 2 and Section 3.2.2). Here level-0 entities are **atomic entities** and level-N > 0 entities are **system entities** that contain at least two lower-level entities, of which at least one is level-N-1.

Following this classification of entities into levels, the domain of interest can be divided into **domain levels**: domain level N comprises all the existing/instantiated level-N entities.

For example, if we characterize bacteria to be level-N entities, then eukaryotes could be modeled as level-N+1 entities, since nuclei, mitochondria, and chloroplasts all originated from ancient bacterial introgressions. Depending on modeling choices, however, these events could either be considered as a single kind of event, leading to the emergence of one higher level N+1, or as separate events, leading to the emergence of several higher levels (Szathmary, 2015). Colonies of bacteria that act as entities in their own right rather than mere aggregates could also be modeled as level-N+1 entities. Multicellular organisms could be modeled as level-N+2 entities, and multicellular organisms in symbiosis with populations of bacteria as level-N+3 entities. Alternatively, one could also think of a model where a human would be merely an environment for gut bacteria, not a separate entity on its own level. In sum, the actual levels identified depend on the model being used, which itself depends on the scientific questions being asked.

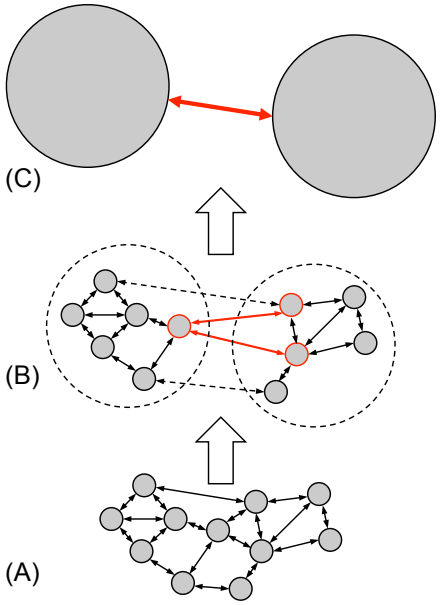

Figure 2: Relationships between the levels of a system. (A) A collection of level-N entities (solid circles) interact (thin solid arrows) in an aggregate system. (B) The dynamics of these interactions stabilize the aggregates into level-N+1 system entities (dashed circles) and these level-N+1 entities contain “specialized interface elements” (red circles) that concentrate most of the interactions between both parts (red arrows). (C) These new level-N+1 entities can be modeled independently of their level N components and the stabilized interactions can be considered as higher-level interactions. Level-N+1 entities may similarly aggregate and interact to form a level-N+2 entity, and so on.

Let us clarify that we do not assume reality to be actually composed of discrete levels. Rather, levels are concepts in our meta-model. A level-N+1 entity is an *abstraction* of a particular collection of level-N entities that have organized and stabilized into a system. Many theories of reality include such levels, either implicitly or explicitly, for example Koestler’s “holonic hierarchy” (Koestler, 1970) in which holons are defined as self-contained wholes that are at the same time dependent parts.

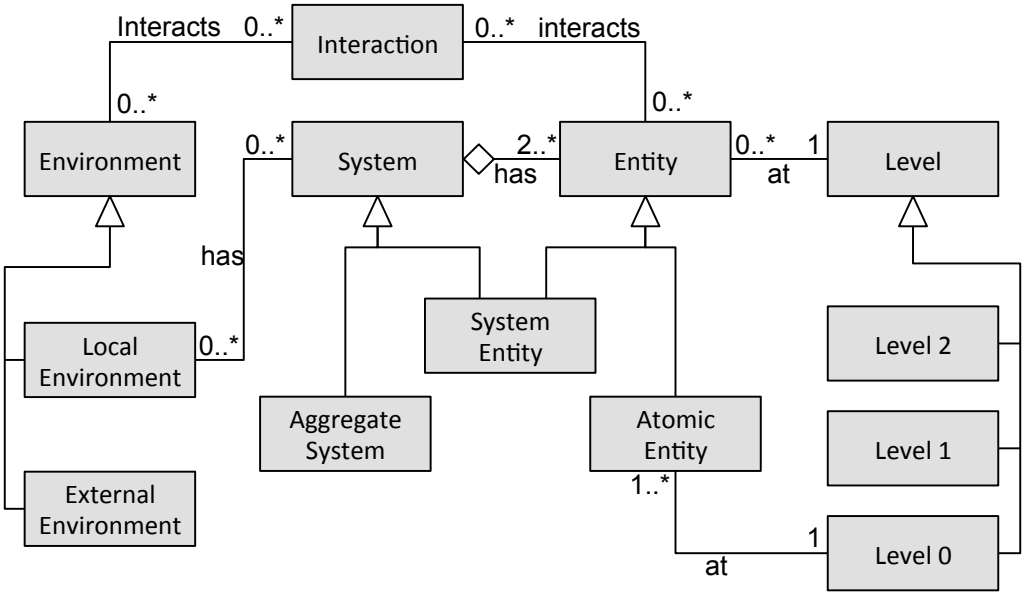

Figure 3: Our meta-model, written in UML, defining the concepts to be used in any model conforming to it. Boxes are classes from which objects in the model can be instantiated. Links indicate associations between classes (associations being named and valued), arrows indicate inheritance ("kind of"), diamonds indicate aggregation. An Environment can be a Local Environment or External Environment. A System can be an Aggregate System or a System Entity. An Entity can be a System Entity or an Atomic Entity. A System has a Local Environment and two or more Entities. An Entity exists at both a system Level and an information structure (model) Level. An Atomic Entity exists at Level 0. Many Environments and Entities can have be involved in each Interaction.

The multilevel model we adopt here permits a level-N entity to have direct interactions with entities on any other level. However, having too many interactions across multiple levels is likely to render the model impossible to analyze. We suggest that, to be useful, a model of an OE system should have the overwhelming majority of interactions between levels occurring between nearest neighbors only: from N to N+1 and N-1.

Putting it all together, our resulting meta-model of environments, systems, entities, interactions, and levels, is shown in Figure 3.

3.2.7. Hierarchies and emergence in the natural world and in human systems

In the natural world, we can observe phenomena that can be modeled as many levels of systems nested within each other. The interest in level hierarchies stems from the fact that they allow us to say something about emergence of complexity in organizations (Lan, 2006). More specifically, we consider higher levels of a hierarchy, as exemplified by living systems in the form of cells, tissues, organs, organisms, groups of organisms and societies, to be the result of processes of emergence. We shall say that the emergence of a new level in a hierarchy is a transition that produces a new system level, which happens in many cases through the composition of groups that subsequently stabilize and form entities on the higher level (see Figure 2). It is also possible for entities to emerge by the loss of interactions, such as with simplification of gene expression pathways in biological systems. But the emergence

of complexity through simplification cannot be the only pathway to emergence, since there is a bound to the number of features that can be eliminated from a system before it becomes trivial.

Niches are conditions of the environment at a certain level that influence the effects of selection on entities at that level. Niche construction is the process by which entities modify their environment, hence improving (or sometimes degrading) chances of survival (i.e. modifying selection forces) for themselves and other entities. Some argue that the process of niche construction is as important to biological evolution as is selection (Odling-Smee et al., 2003). While we do not want to engage in that discussion, we note that the active modification of the environment is a process that can be observed at several levels in the biological hierarchy as well as in social systems and other artificial hierarchical systems.

If one considers the emergence of "tentative" groups of entities to be the first (haphazard) step in the emergence of a new level, it follows that only those groups that achieve a certain degree of coherence can become candidates for entities at the emerging higher level. It is through downward causation[4] constraining the behavior of lower-level entities that coherence becomes more pronounced. Competition of gradually more coherent groups of entities sets in and further channels the variations provided by the lower level, building a new System Entity from patterns in the Aggregate Systems.

We exemplify our view by discussing a specific case, Darwinian evolution, in the general model (Figure 4). Any model of Darwinian evolution would include a population of biological individuals ("organisms") at a level N. These individuals are entities having the property of reproduction (sexually or asexually). Now, reproduction is governed by molecular mechanisms at the lower level (N-1). Since these mechanisms are error-prone, level N-1 drives the variation component of Darwinian evolution. At the upper level (N+1) the individuals are organized in populations that interact in a finite world. These interactions result in a differentiation of the reproductive success (due to Malthusian mechanisms, predation, parasitism, altruism, etc.). This may result in many selection types, such as a neutral process (if the population is too small regarding the difference in reproductive success), natural selection in the Darwinian sense, or removal selection as in artificial systems. Ultimately, the interactions of these three levels (molecules, individuals, population) result in the realization of a Darwinian process.

[4] We circumvent the question of whether or in what sense "downward causation" exists in reality by focusing on models where we introduce it as existing.

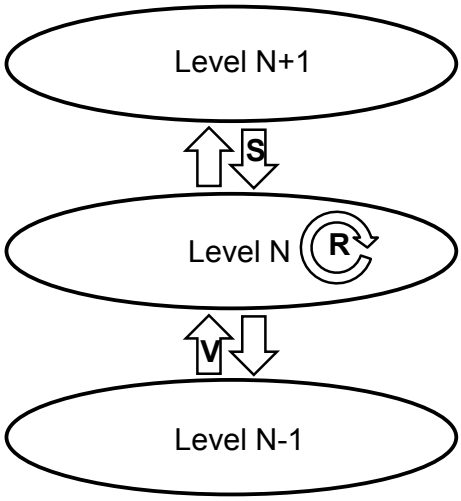

Figure 4: A general model of three levels of an evolutionary system interacting within and between levels. Level N is the evolving layer: it is composed of entities having a (self-)replication property ("R"). Selection ("S") is a constraint imposed by the system at level N+1 (the "ecosystem", which may include level-N+1 entities such as a public health agency) upon level N. Variation ("V") is a consequence of the laws of level N-1 (molecules, including DNA, in Darwinian evolution) on the behavior of level N. This general model encompasses both the scientific model of Darwinian evolution, and engineering and simulation models (though these may include "shortcuts" implemented at any of the three levels; see below).

In socioeconomic (or, more generally, socio-ecological) systems as well, the emergence of new entities may be identified with the emergence of downward causation, or agency as it is referred to in the social sciences. Individual people have agency, and are thus entities, but collections of people are often simply queues or crowds. But when their behavior in a group is formally constrained so that it is the group itself rather than the individuals composing it that exercises agency, then the formally structured group is itself a higher order entity such as a corporation, a government, or a regulatory agency; it is able to exercise causation on other entities at its level or below. As was pointed out in section 3.2.6, there is no clean separation between levels, but in the case of socioeconomic systems the separation may be not just model-dependent, but intrinsic. In Western societies individuals are legally treated as autonomous entities. But many individuals are also "components" of higher-level entities such as corporations or government agencies, and in that context must act out the appropriate role, which is not one of an autonomous individual.

The situation in human systems is even more complex because some entities in these systems, specifically governments, have acquired or given themselves the power to confer entityhood, (for example, when a business is incorporated), and the consequences can be unusual. For example, if we consider the atmospheric system in isolation, then it appears to be one in which macro-scale patterns such as storms emerge, much like the flocks in boids (Reynolds, 1987). Yet as previously pointed out, these do not constitute new entities and therefore no new, higher level is present. But if a storm satisfies the legal definition of a hurricane, then from the point of view of a governmental entity the storm itself becomes an entity, with a name, and its presence in the area causes the release of reconstruction funds

that would not have been available if the storm had not achieved the status of an entity. The storm itself as a physical phenomenon may cause changes in coastal land forms, but the storm as a legal entity can induce other changes in the local geomorphology, such as dykes and drainage canals built with the reconstruction funds. Thus, depending on the point of view, the storm both is and is not an emergent higher-level entity, and these points of view are inherent in the system itself.

3.2.8. Information

Life provides the dominant template as well as the language for thinking about OE. A key marker of the transition from non-living to living systems is that, unlike non-living systems, living systems are information-dependent. They are dependent on information because they include as an essential component explicit models of themselves. Indeed, life may be defined as an entity that has a model of itself and its relations with its environment (Rosen, 1991). The genome may be considered to be analogous to the code describing the model, while the phenotype in a sense executes it in order to survive and reproduce.

With more complex organisms such as vertebrates, the models are not just coded 'genomically', but also 'neurally' based, and in human systems some are coded 'linguistically' in oral traditions and even in external devices such as books, pictures, and software. The fact that all living systems, including economic and social systems (and the technology produced by these), incorporate models in an essential way means that they experience an additional source of indeterminacy. This indeterminacy arises from the fact that the models are necessarily incomplete or imperfect, and their imperfections are various, depending on the particular circumstances under which the models were generated, while behavior of these systems depends to a greater or lesser extent on the various models held by the individuals making up the system. This form of indeterminism associated with "knowledge creation" (i.e. model creation) was the crux of Popper's (1982) argument for an "open universe". One manifestation of this indeterminacy is the ability of agents to mislead or cheat other agents – or themselves.

Furthermore, because the models are created in part on the basis of perceptions of the surrounding environment or system, living systems have evolved to a degree on the basis of their own perceptions of themselves. Perception must therefore be considered a fundamental element in evolution and OE[5]. Perceptions and misperceptions, and the model errors engendered by misperceptions, may, in some biological systems, rival genome-level phenomena as a source of the variations required for evolution; in the evolution of social systems they are clearly of primary importance. Finally, perception always involves a point of view. Therefore, in all living systems, "point of view" is an intrinsic part of the phenomenon we are trying to understand – not just something we have as we look at the problem of open-endedness and decide what approaches we want to take in trying to understand it.

[5] By "perception" we mean the ability of the System to sense (by whatever means) aspects of its Environment, allowing it to act and react; it would not necessarily need to be a living System.

3.2.9. Timescales

We adopt the following notion: one way to distinguish levels is by the timescales on which their dynamics can be observed.

A natural question to ask is what provides the conditions for a level in the model and how we can recognize these conditions to be fulfilled in a particular hierarchical system we want to observe. Since the systems we intend to consider are open and dynamical, the notion of time is an indispensable element of our discussion. Higher levels typically have slower dynamical timescales associated with them, as when groups of organisms forming a society (higher level) develop on a slower timescale than the organisms (lower level) themselves. The difference of dynamical timescales between levels can be many orders of magnitude, but might also be down to few orders of magnitude only, with extreme cases at the lower end of perhaps just one order of magnitude. When such a difference in dynamical timescales is relevant, higher levels need to slow down their development in order to exist or be perceived by an observer as something different from their elements.

In cases such as these, we might say that downward causation from level-N coordinates the level-N-1 entities. We might also say that the emergence of a level-N+1 entity is due to regularities and coordination at level N. Hence the level we focus on helps determine our explanation of the various levels.

4. Open-Endedness and Novelty

We use our meta-model, defining entities, systems and levels, to consider OE more closely, starting from the definition already mentioned:

Open-endedness is the continual generation of novelty.

Given this definition, there are different kinds of OE depending on the kinds of novelty generated. We consider novelty in more detail here.

Once the relevant domain of interest has been modeled using the meta-model in section 3.2, and simulated, one can then use the model in order to interpret observations of its dynamics and to identify different kinds of novelty and observe different kinds of open-endedness. It follows that the kind of novelty/OE identified is relative to some model and meta-model of the system under investigation.

In this Section we define different types of novelty, then discuss the different classes of OE to which they can lead.

4.1. Types of Novelty

Having defined the primary concepts of consideration in the meta-model, we now turn to dynamics. Due to their interactions, the systems and entities change over time. These changes can be observed and described in the model of the system as changes of the entities. Since level-N entities are composed of lower-level entities and are components of higher-level entities, a change observed at level N is generally not confined to a single level:

it may be caused by changes at lower levels and may itself cause changes at adjacent levels[6].

When focusing on a given change in the system, three cases can be distinguished, depending on whether the observed change can be described within the given model, within the given meta-model (but with a model change), or whether it requires a change to the meta-model. These three cases lead to three different types of novelty:

4.1.1. Type-0 novelty: variation

Variation: novelty within the model.

Variation is a change to an instance of the model, a change to the values of a variable that conforms to the model.

Variation explores a pre-defined (modeled) state space, producing new values of existing variables as in, for example:

- changing the value of an integer-values variable: $x = 4$, then $x = 5$,
- changing a gene to a different allele,
- increasing or decreasing the size of a finch beak,
- changing the number of individuals in a population (including to zero: extinction),
- swapping out one entity for another of the same type,
- flipping a bit in GA with fixed length bitstrings,
- changing prices of existing economic goods.

In our meta-model, a variation does not change the set of entities the model captures at level N. Thus, the combinatorics of these entities, i.e. the statespace, is not changed either.

4.1.2. Type-1 novelty: innovation

Innovation: novelty that changes the model.

Innovation is a change to the model: a change that adds a new type or relationship that conforms to the meta-model, or possibly eliminates an existing one.

Innovation changes the combinatorics and the size/structure of the state space, thereby growing/shrinking the possibilities of variation. Examples of changes to the model include:

- moving the value of a variable outside the range constrained in the model: x is modeled as an integer value initially, then $x = 0.5$,
- adding a new dimension of the same type: $x,y : X$, then $x,y,z : X$,
- duplicating a gene or chromosome,
- growing a parse tree in Genetic Programming,
- adding a new species in an ecosystem,
- adding a new product or production technology in an economic system.

[6] It may also have no effect at all, as, for example, is the case with a neutral mutation.

A question for our specific meta-model is whether an innovation systematically corresponds to an increase of the number of entity types. Innovation may correspond to the addition of a new species, but this event may be associated with the extinction of other species at the same level, thus reducing the total number of entities at this level. Examples of such a process can easily be found in the economy when a new economic good replaces a previous set of goods (for example, the computer replaced the typewriter and the mechanical calculator). Note that the number of entities cannot be reduced continuously at a given level.

The extinction of a species that provides necessary ecological services is a fundamental change to the ecosystem, and can be modeled as a type of innovation at the ecosystem level caused by the loss of a species. However, there is no necessity to remove a type from a model when that type has no instances. If the model is not so modified, the extinction is classed instead as variation, with the value of the population size variable decreased to zero.

4.1.3. Type-2 novelty: emergence

Emergence: novelty that changes the meta-model.

Emergence is a change to the meta-model: a change that adds a new meta-type or relationship, or possibly eliminates an existing one.

Potential examples of emergent phenomena include:

- the addition of the concept of discrete entities to a meta-model based on continuum concepts,
- the addition of a new mechanisms for variation, such as sexual recombination,
- the addition of the concept of processes to a meta-model based on entities,
- the addition of the concept of energy to a meta-model based on movements.

One particularly important phenomenon is the emergence of a new level of organisation. Our particular meta-model has been designed to highlight this form of emergence, by the inclusion of the concept of discrete levels.

In our meta-model, a change observed at level N cannot be described at this level because it changes the composition rules of the level-N entities. Thus, to be described in our modeling language, this change needs the creation of new entities at level N+1 or the elimination of all entities at some level. Note that this can require a reorganization of the levels structure. This is the case, for example, when bacteria and a multicellular eukaryote engage in an endosymbiosis relationship: if the multicellular eukaryote is level-N, the compound is level-N+1 and the ecosystem, formerly level-N+1, is now level-N+2. Or, if the loss of a species at level-N causes the loss of the ecosystem at level-N+1, then any structure at level-N+2 becomes level-N+1.

This generally happens when a population of entities (aggregate system) becomes a new system entity at a higher level. Examples are:

- when configurations in cellular automata exhibit large-scale coherent patterns (see section 7.1),

- when bacterial cells evolve into eukaryotic cells with cytoplasm, nuclei and organelles,
- when single-celled organisms form organized multi-cellular structures such as animals, plants and fungi,
- when solitary individuals become colonies or societies,
- when a population of cell tissue forms an organ,
- when a group of individual economic agents pool their resources and form a company,
- when a number of species form an organized ecosystem,
- when a number of artists form a new school.

We have specifically included each new level as a separate component in our meta-model, ensuring that each transition to a new level is a type-2 novelty. Consequently, each new level will be added to the meta-model only once, and emergence corresponds to the first occurrence of the type of entity. Once the first level-N+1 entity has emerged as a type-2 novelty, similar entities appearing later will be added to a pre-existing level. Thence, the next level-N+1 entity to appear is only a type-1 novelty, an innovation at level N+1. Indeed, it "simply" changes the level-N+1 structure, but no further change to the meta-model is needed.

4.1.4. Classification versus measures

We have provided a classification of novelties, with respect to some model and meta-model. Our classification complements various measures of novelty in the literature, for example, those of (Bedau and Packard, 1992; Bedau et al., 1998; Droop and Hickinbotham, 2012). Those measures tend to be expressed in terms of 'components' (be they genes, individuals, or whatever); they track abundances of different components as new components enter the system. It is assumed that it is possible to distinguish and compare these new components.

Our classification allows us to determine if the components differ through variation (if they are mutated strings, say), or through innovation (if they are new species to the model, say), or differ through emergence (if they are new levels of organization, say), dependent on the model in use (in the current literature, the relevant model is usually implicit). The cited measures could then potentially be used to quantify the respective amounts of variation, innovation, or emergence activity, in order to determine if the activity of the respective type is continuing (see also section 4.3).

4.2. Events and classes of events

Any particular change is potentially the result of a set of changes at a range of levels, and may result in changes at a range of levels. We group such changes into an event: a set of cascading changes at various levels of the system that ultimately corresponds to a single novelty at a given level. It directly follows from this definition that events can be classified into three classes according to the highest type of novelty they generate at the different levels where they are observable.

The distinction between changes (observed at a given level) and events (groups of changes at all levels) is a fundamental one. In particular it captures the question of timescales (see

section 3.2.9) since a single change at one level may be the consequence of many changes at the lower levels. For a given system, the observed events will be relative to the model and meta-model. The composition rules that are used to define levels in our meta-model have no equivalence for the types of novelty: any type of novelty at a given level can be the cause or consequence of any type of novelty at another level. In particular, all kinds of novelty can ultimately be connected to a set of variations at level 0.

4.3. Open-Ended Events and Open-Ended Systems

We are now able to discern three types of events in our model: *variation*, *innovation* and *emergence*. Two of these types, innovation and emergence, we define as *open-ended events*.

Open-ended system: a system with the ability *to continually produce open-ended events*

Variation (type-0) events correspond to the "normal" regime of any dynamical system, and thus we do not consider them open-ended. For example, engineered mechanical systems (for example, a car) undergo events, but none of these events change the structure of the system (i.e. number and kind of entities composing the car).

Innovation (type-1) provides a form of OE of new kinds of things (new types in the model), that nevertheless corresponds to existing concepts (the new types conform to the concepts in the meta-model). Emergence (type-2) provides another form of OE, of new concepts (new components in the meta-model). We have designed our specific meta-model to support the idea of "major transition" although this idea is not made explicit here. According to (Maynard-Smith and Szathmary, 1995) a major transition is the emergence of an entity composed of (sub-)entities that were capable of autonomous replication before the transition but that are no more capable of replication after – while the aggregate entity is now capable of replication. In our meta-model, a major transition is a type-2 open-ended event (emergence) that adds a new level, and therefore the ability to evolve on a higher level. However, the loss of the replication property at the lower level is not included in our meta-model because this property is still present in the low-level entities that are not engaged in this specific major transition. Different changes to this meta-model (adding other concepts than new levels) would probably lead to other classes of emergent novelty.

Interestingly, the distinction between type-1 (innovation) and type-2 (emergence) events, captures the distinction between the notion of continual novelty and the notion of continual increase of complexity. The repeated occurrence of type-2 events does not however imply that there is an arrow of "progress". A type-2 event implies that new levels must be added to the model, but these can be the cause or consequence of a decrease of "complexity" at other levels through partial or complete loss of structure or function. For example, the establishment of an endosymbiotic relationship between bacteria and multicellular organisms can lead to a heavy loss of complexity of the bacteria (Batut et al., 2014; McCutcheon and Moran, 2012).

5. Limits to Open-Endedness

Proving that a system will continually produce open-ended events is challenging. On a theoretical basis, one can question whether such systems, when instantiated physically, are even possible in a finite universe. A continual production of novelty would require physical systems that are unbounded in space, time, and combinatorial possibilities. Conceptual models and other abstractions are not limited in this way.

Should we be looking for systems able to continually produce open-ended events, or "simply" for systems able to produce a sufficient number of open-ended events? We thus distinguish systems that are theoretically open-ended from those that are effectively open-ended. The former may be demonstrable in a mathematical universe, but questionable in a finite universe; the latter are questionable in a mathematical universe (at least for non-platonists), but may be demonstrable in a physical one. If one says that life is open-ended, one can only claim its open-endedness so far. Similarly, when seeking to simulate open-endedness most authors are willing to simulate one or a small number of open-ended events. This point is further discussed in section 5.3.

5.1. Upward Limits

There are severe limits to achieving open-endedness in simulation and also in the real world. The growth of resources required to populate the higher levels will quickly lead to an exhaustion of material or entities. In biological systems, it was this Malthusian limitation that led Darwin to develop the theory of Natural Selection.

A requirement of any such system is a lower level that comprises entities with combinatorial power and behavior, which can generatively combine in multiple ways to form a huge ("effectively unbounded") diversity of entities. We call these entities, and the rules for their combination, the "chemistry" layer. This is level 0[7]. Suppose that, at each level i, m_i entities combine into a single higher-level system, and every lower level entity belongs to exactly one higher-level system[8]. A level-1 system comprises m_0 level-0 entities. A level-2 system comprises m_1 level-1 entities, each comprising m_0 level-0 entities, hence it contains $m_1 \times m_0$ level-0 entities. A level-k system contains $\prod_0^{k-1} m_k$ level-0 entities. If we assume for the sake of the argument that all the m_i are identical (and that entities' components do not overlap and are not shared), then such a level-k system contains m^k level-0 entities.

Hence the number of base entities needed to populate a system grows exponentially with the number of levels. After several levels, a system like this will exhaust the number of level-0 entities available (molecules, computer memory cells). Moreover, for a given m_0 number of atomic entities at level 0 the maximum number of entities at the highest level rapidly decreases as the number of levels grows. As long as Darwinian selection is at work, this

[7] With our meta-model model, it is not "turtles all the way down"!

[8] This does not imply or require that the set of m_i entities constituting any specific level-i+1 individual is fixed or static for the lifetime of that individual. It is a specific characteristic of biological individuals that they continuously turn over their constituent lower level components, while retaining their systemic coherence and individuality.

decreases the efficiency of selection at this level and can increase the amount of "drift" (Kimura, 1984). A direct consequence is that the possibility for the system to acquire a more complex structure (innovation events) also necessarily declines as the number of levels grow. Another consequence is that innovation at each level will be less constrained by selection at that level, and more the consequence of intrinsic dynamics such as drift.

To summarize, the open-endedness of any discrete system with $m_i > 1$ will be limited by the combination of two processes: first, the exponential exhaustion of level-0 entities used in the system's "complexification" limits the possibility of observing type-2 events (emergence); second, the reduction of the selection efficiency with the decrease in number of entities at the higher level limits the possibility of predicting the direction of type-1 events (innovation). Therefore, one can conjecture that a finite discrete system cannot be theoretically openended (in the sense that it can reach barriers to further novelty that could be transcended only through some "external" intervention to "expand" the finite limits).

5.2. Downward Limits

Schrödinger (1944) hypothesized that living systems were only possible on the macroscopic scale, due to quantum effects at the atomic level. More generally, he argued that order on large scales was a consequence of disorder on small scales, which he called the "order from disorder" hypothesis. For him, living matter must display homeostasis, which is maintenance of large-scale (relative to the atomic scale) organization over time. But he also pointed out that living things inherit and bequeath order along their lineage, and that the mechanism for maintaining order through lineages must be small, on the molecular scale. He even postulated that this material was probably an "aperiodic crystal" that conveyed information via covalent bonds. Inheritance on the molecular scale would be a possible explanation of hereditary variation arising, ultimately, from quantum effects. Schrödinger's breathtaking hypothesis prefigured the eventual discovery of the biological function of DNA by Crick and Watson a decade later, and both credited him as an inspiration for their work.

For our purposes, Schrödinger's foray into the molecular basis of life reminds us that innovation is only possible in physical systems that are sufficiently large. If living things were too small, quantum effects would make inheritance impossible: it would be impossible to maintain structure, and innovation is typically only possible in physical systems that are sufficiently large (Anderson, 1972).

5.3. Effectively Open-Ended

As explained above, the hierarchical organization of our meta-model imposes strong limits to the open-endedness of any bounded system, particularly when one considers type-2 open-ended events (emergence): the number of levels one can model from a finite number of level-0 entities is severely limited by a potentially exponential increase of level-0 entities dynamically linked in level-N entities. Moreover, if one simply considers variation in a state space with a finite number of dimensions, each of which can take a finite number of values (for example, a bitstring which forms a boolean hypercube), then the state space itself is finite and bounded, and so the possible variations are bounded. Strictly speaking, the system is not open-ended. However, considering a system S at a time t of its evolution, we propose

to say that S is effectively open-ended if the size of the population at the highest level of the hierarchy is large enough such that open-ended events are still possible.

Physical systems (for example, the biosphere, techno-social systems) are effectively open-ended if one considers type-1 open-ended events (innovation). Indeed, these systems have a sufficiently large state space and the states realizable or explorable within a reasonable time (say, within the age of universe) are an insignificant fraction of the entire set of "interesting" states. When considering type-2 open-ended events (emergence), whether these systems are effectively open-ended is for us quite literally an open question. Indeed, either the biosphere or the techno-social systems are now limited by the finiteness of earth's resources and the possibility of the emergence of new levels of organization is not clear[9].

Now, consider computational simulations[10]. The number of entities in a simulation is severely limited by computational power, including both CPU time and memory. In the universe, there is a vast number of molecules; in a computer simulation, there is a large, but not vast, number of memory cells (its "effective unboundedness" is less effective). If we wished to simulate interesting models with many levels, we could not do so from the bottom up: there would simply be too many level-0 elements to simulate efficiently, or even to fit into memory. Additionally, regularities in the dynamics of a level-N+1 entity comprising a system of level-N entities tend to be on a longer dynamical timescale (slower processes) than in the level-N entities themselves and the events we are seeking are likely to be rare events. This implies the need for an exponential amount of simulated time in a multilevel system. Thus the upward limits will be quickly reached as the number of levels grows (that is, as emergence events accumulate). One can still simulate effectively open-ended systems if the total number of levels in the system is low enough for the number of entities to be large. However, though effectively open-ended, the system will eventually exhaust its possibilities.

6. Simulation of Open-Endedness

Consider artificial life as an example of computational simulation. One of the central goals of ALife is to use simulation to study and understand open-ended systems, including open-ended Darwinian evolution (Bedau et al., 2000). In particular, we wish to study how living systems have undergone successive major transitions from the first replicative molecule to the techno-social level (Maynard-Smith and Szathmary, 1995). Systems of this kind have a hierarchy of levels, with lower-level entities forming higher-level systems. Thus this goal directly hits the computational limits we conjectured above and may be an inaccessible dream, at least with current computational technologies. Hence we need to optimize the approach, taking stock of the exhaustion of resources. What that means is that we need to

[9] One can still argue that, on a larger horizon (namely on the level of the entire universe), although the universe is bounded, the potential for novelty is unlimited, be they generated by variation, innovation or emergence. Following this idea, one could consider that the universe is effectively open-ended.

[10] As opposed to statistical simulations, such as "draw an infinite number of reals from (0;1), with an exponential distribution", where innovation in the form of seeing new numbers is certain. Our argument here applies to any physical simulation, not just to computation alones.

hard-code some of the emergent properties and laws at certain levels, rather than letting them emerge from lower levels. Some hard-coding will cover the entire set of lower levels not considered (the "generative" layer), and some will cover the examined (emergent) levels to "cheat" (optimize, hard-code) certain behaviors through what we call shortcuts.

6.1. Generative Layer

The question of level-0 entities in natural systems is an open-question: is level 0 populated by molecules, atoms or even lower-level particles? Now, in a simulation, one cannot start from too low a level, simply because it would naturally increase the number of levels that separate level 0 from the level of interest for the question under study. Thus, our multilevel model will generally possess a lowest combinatorial level (level 0), which we refer to as "chemistry" or "artificial chemistry" (Banzhaf and Yamamoto, 2015; Dittrich et al., 2001). This level should be able to subsume all the diversity created by putative lower levels (atoms, particles, and below) and, conversely, provide the necessary diversity of elements to the higher levels. This basal level provides the "physics" (rules and behaviors) and "chemistry" (combinatorics) of the simulation, thus implementing level 0. For example, in biological evolution, the existence of elements that can be combined through bonding dynamics into molecules of different types – in itself a substantial compression of what goes on in atom/quantum physics and quantum chemistry – provides different chemical characteristics as a prerequisite for the variety we can observe at higher levels of the hierarchy.

Examples of generative layers are numerous in the ALife literature. It can be composed of bit strings as in the case of genetic algorithms (for example, Holland, 1975), artificial molecules in artificial chemistry (for example, Hutton, 2002) or code in genetic programming (for example, McMullin, 2012), to give a few examples.

6.2. Shortcuts

If one wants to study a layered system like the ones we envision in our metamodel, one needs to simulate multiple levels, that successively emerge from the generative "soup" of level 0. As conjectured above, such a simulation is impossible to implement in practice for large values of N. We may thus need to introduce new elements in our simulation that will simplify or accelerate the dynamics of the higher levels. We call these new elements shortcuts since they directly implement some properties of the higher level that, in an ideal simulation, would emerge from the generative layer.

Shortcuts are hard-coded design optimizations manually introduced into the simulation. One can consider the shortcuts as "cheats", which limit the generality of the model. It would certainly be preferable to be able to simulate open-ended systems (or at least effectively open-ended systems) without making use of any such stopgap measures. This is clearly feasible, but the number of emergent levels in the simulation will definitively be limited and this would probably preclude the simulation of deep multiscale models[11], such as those ranging from molecules to ecosystems. If one wants to simulate multilevel systems with a large enough number of levels, shortcuts will be necessary and, to the best of our

[11] A two scale model is technically "multiscale", but can perhaps be simulated in some cases.

knowledge, all ALife simulators designed and implemented so far have used some kind of shortcuts. It is better to be aware of this fact and to use it explicitly, rather than to introduce shortcuts implicitly in a simulation without even realizing it.

The problem of how to try to simulate OE can be reformulated as: How to shortcut in a principled way? or, How can we design shortcuts that provide optimizations, without precluding the possibility of particular kinds of OE of interest. Indeed, shortcuts provide optimizations by explicitly constraining structures and behaviors at their level, rather than requiring these constraints to emerge from the system's behavior. Hence, specific shortcuts will enable or constrain certain classes of OE.

When designing the simulation of an open-ended system, one crucial design step is then the identification and implementation of relevant shortcuts. These will be research-dependent, i.e they will depend on the specific question one wants to answer with the simulation. Examples of such potential shortcuts that might be inserted in certain simulations include individuality, replication, and fitness.

6.2.1. Individuality

Individuality is the "mother of all shortcuts". Entities of a level N > 0 are hard-coded rather than emergent. This can be implemented in two slightly different ways: either by directly simulating the dynamics of these entities, which could be called an "identity shortcut"; or by setting boundaries that group together entities of level N-1 in level-N entities, which would rather be a "boundary shortcut". In Figure 2, an identity shortcut hard-codes the circles of subfigure C while a boundary shortcut hard-codes the dashed circles of subfigure B.

Obviously, this shortcut makes it impossible to observe emergence at level N but it still enables emergence at level N+1. However, when limited to the boundary part, an individuality shortcut enables innovation at level N.

6.2.2. Replication

Replication is a central process in biological evolution. In computational simulations, a replication shortcut is the explicit implementation of operators that replicate the entities at level N from the outside of the entities. This shortcut generally comes with secondary ones, for example, to synchronize the replication of all the entities of the same level or to include variation operators that slightly modify the offspring.

6.2.3. Fitness

Fitness in evolutionary computation is a classical shortcut. It replaces the differential reproductive success emerging from the difference between level-N entities (in terms of composition of level-N-1 entities), and from their interactions in the level-N+1 population, by an explicit computation of reproductive success according to some target task such as computing a given function, seeking "food", gaining "energy", and so on[12].

[12] Fitness in biological systems is defined as differential reproductive success. But in nature fitness is assessed retrospectively, through natural selection.

Many authors have argued that explicit fitness precludes OE (Baptista and Costa, 2013; Channon and Damper, 2000; Ray, 1992) and some have proposed to replace it by implicit fitness or by subtler approaches such as selection for novelty (Lehman and Stanley, 2011; Soros and Stanley, 2014). This is sometimes merely a change in terminology, renaming the target task with something more biologically sounding, such as replacing "increasing a score" by "accumulating energy".

However, OE should be possible with an explicit fitness at a level N (which would shortcut differential reproductive success driven by interactions at level N-1), providing that level N entities interact in such a way that level N+1 can still have an influence on their reproductive success. In effect, the variations at level N (the phenotype) may induce variation at level N+1 (the ecosystem). The fitness values of level N entities, though explicit, will then change due to the downward causation of level N+1 on level N (that is, the same phenotype could have different fitness depending on its neighborhood). This could happen, for example, when the model includes an explicit fitness for resource uptake but that this resource is shared by all individuals. This is, for example, the case in Tierra (Ray, 1992).

6.3. A General Architecture for Open-Ended Simulations

In Figure 5 we identify a general architecture for layer-based emergent simulations of OE. This architecture separates out the base layer, implementing the physics and chemistry, from the shortcuts at higher levels. The base layer of "generative rules" provides the implicit coding of all levels below the lowest simulated level. These rules generate and enable the lowest level of simulated entities, and the relevant lowest level of an internal environment.

In addition, each simulated layer may have some hard-coded shortcuts. The collection of these is the set of "structuring rules". These rules provide optimizations by explicitly constraining structures and behaviors at the level, rather than requiring these constraints to emerge from the system's behavior. In particular, these rules might provide shortcuts for determining or recognizing the identity of entities, for communicating between entities at a level, for providing parts of the internal environment, and for explicitly implementing "downward causation"-style constraints imposed by higher levels of the system.

Note that, for population systems in this architecture to be at a higher level than their constituent entities, they cannot be mere aggregate-systems (which do not have a level in our meta-model) but must be genuine system entities. If we were to consider a population-as-aggregate system to be at a higher level than its constituent entities, then we could never get interesting emergence (in this meta-model[13]). This is because we would generate the higher level (and get emergence) merely by virtue of having the population; there would be no opportunity to generate the (emergent) new level later, when the higher-level system

[13] This is another possible difference between our computer simulation motivated metamodel and biology. This argument in a biological context would imply that innovation in living systems is impossible without group selection. This would be a highly contentious claim.

entity itself arose. Making a population into a system entity immediately is a form of shortcutting an emergent transition.

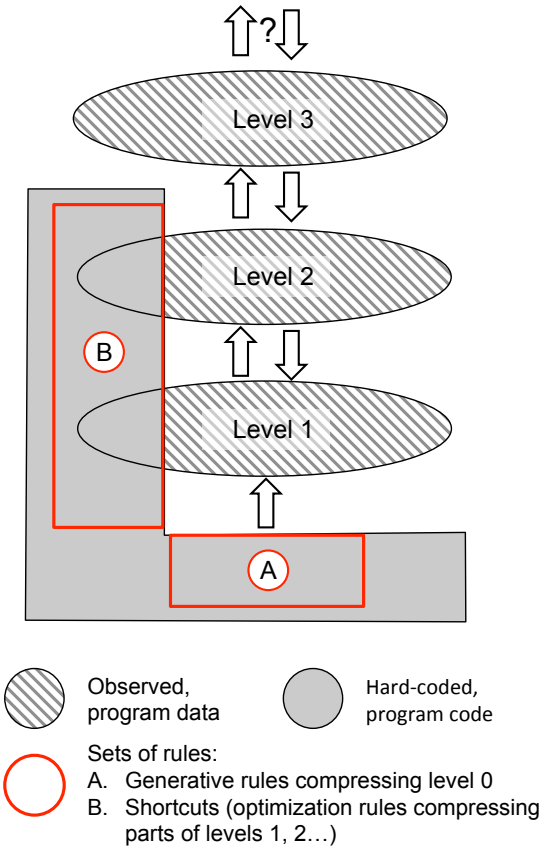

Figure 5: Simulation with shortcuts, generic figure. Levels 0 is abstracted by the set of generative rules encoded in "A". Levels 1 and 2 are predefined by the simulation (parts of them are hard-coded) but they include emergent parts that enable variation, differences or innovation. Level 3 is fully emergent, not at all included in "B" (but may be anticipated by the modelers when designing "A" and "B" since they want to observe 3). If level 3 is a system made of entities of lower levels, then its emergence is a transition. If the level 3 entities are replicators, then emergence of level 3 is a major transition.

6.4. Changing the Model

The previous architecture provides a methodology to simulate open-ended systems by means of software components that are not open-ended. However, if one wants to use OE to create software technologies in order to implement creative machines, a.k.a. computational "living technologies", one may want to implement OE directly at the software level. But how can changes working outside the model be implemented in software? In conventional programming languages, the code is fixed, so the design model is fixed. All that happens is that the pre-existing code executes. Generally, no new code is produced. In some languages, however, it is possible for code to modify itself as it runs[14]. In standard software

[14] This is different from external modification, or "patching", running code.

development, self-modifying code is generally to be avoided, as it makes it difficult if not impossible to know beforehand what the code will do; and therefore very difficult to design (or debug) code to demonstrate specified behaviors. Developers usually write code to do something specific and well-defined (such as calculate payroll, control power stations, or launch small irate birds at acquisitive pigs). In low-level assembly languages, self-modifying code is relatively easy to achieve. “Automata chemistries”(Tierra, Avida, Stringmol) are examples from ALife that exploit this concept, constructing entities that are strings of assembly language instructions, hence new code (see further discussion in section 7.5). Some higher-level languages have also been designed explicitly to facilitate program self-modification (Spector and A. Robinson, 2002).

In most high-level (structured) languages such as C++, Java, Python, self-modification requires special effort and is not possible in all cases (other than using the language to implement an interpreter for a new language in which it is possible). A “reflective” language can examine its own code; some languages (particularly interpreted rather than compiled languages) can also produce new code on the fly. Such a language is needed to implement software that can change its own model as it runs. The final model of the system, implicitly produced by executing the software, will need to be inferred.

6.5. Capturing the Novelties

The shortcuts and corresponding simulation code above explicitly implement the (computational) model of the simulation. In order to capture novelties that change the model (innovations) and meta-model (transitions), there needs to be simulator code to make these changes. There are several possibilities:

1. The emergent novelty is recognized *outside the simulation*. The simulation data is analyzed and the presence of relevant emergent “model-breaking” phenomenon is inferred, but not explicitly realized as a new entity or level in the simulation (for example, the emergence of a flock in a boids simulation (Reynolds, 1987), recognized through visualization by the observer or by statistical analysis, but not explicitly present in the simulation code).
2. The type of emergent novelty is *anticipated and recognized*, with pre-coded recognizers present in the code, available to report it once it appears (for example, a boids simulation with a pre-coded flock-detector, that recognizes if and when it emerges).
3. The type of emergent novelty is *anticipated and captured*, with pre-coded structural rules present in the code, available to recognize and make it a component of the simulation once it appears.
4. The emergent novelty is somehow *emergently recognized* by the simulator, and new code is generated by the simulator to capture the recognition – for example, self-modifying code (Stepney and Hoverd, 2011).
5. The emergent novelty is somehow *emergently captured* by the simulator: once recognized, new shortcut code is generated by the simulator to capture the specific emergence (for example by self-modifying code). The difference to the previous situation is that the new shortcut here is not only detected as an event but shortcut

code is also created that accelerates the simulation by explicitly coding the new, emergent, level.

6.6. Design Principles

We can use the CoSMoS (Complex Systems Modeling and Simulation) approach (Andrews et al., 2010, 2012; Stepney, 2012; Stepney et al., 2016; Stepney and Andrews, 2015) to help implement a principled computer simulation that conforms to the architecture described above. CoSMoS is a particular approach to simulation of complex systems and their emergent properties, but the components are necessary (if not always explicitly identified) in any principled simulation work. The components include:

- **Research Context:** a description of the research question(s) to be addressed by the simulation, for example, “to understand the system’s exploration of its designed search space”, or similar.
- **Domain Model:** a scientific model of the biological, socioeconomic or other system of interest, including the emergent levels and OE classes of interest (for example, speciation, or a major transition, in an evolutionary model), such as shown in Figure 2 or 4.
- **Platform Model:** an engineering model of the software simulation system. It does not include a model of desired emergent properties, explicitly ensuring that the answer is not coded into the software. It includes, however, additional features needed for efficient simulation, such as the generative and optimization shortcut components in our proposed architecture (Figure 5), and interfaces.
- **Simulation Platform:** the software implementation, built from the Platform Model specification, suitable for running Simulation Experiments.
- **Results Model:** a scientific model analyzing the output of the simulation experiments in terms of Domain Model concepts.
- **Argumentation:** that the simulation is “fit for purpose" relative to the Research Context; specifically here, that the shortcuts neither inhibit nor hard-code the desired OE in the simulation platform.
- **Meta-Model:** a model encompassing the concepts used in the Domain, Platform, and Results Models, to ensure consistency between these concepts across the process.

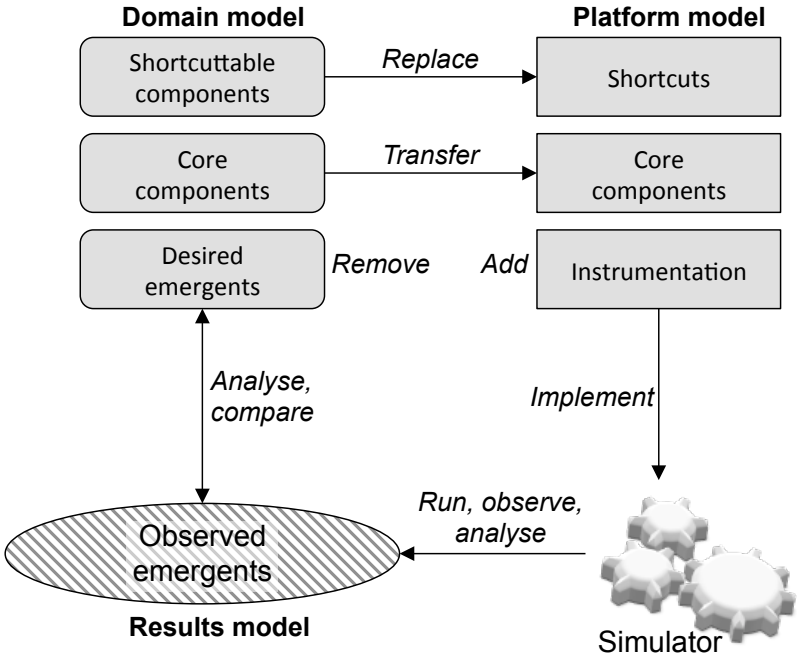

Figure 6: Open-ended novelty simulation design within the CoSMoS approach. The Domain, Platform and Results models are instantiated from a common meta-model, and related to each other as shown (see text for further explanation).

Within the CoSMoS approach, the framework described in this paper gives a structured way of thinking about the design of a system to display open-ended novelty, including the model, any simulation of it, and the effect of particular design decisions.

For example, we could design a simulation to investigate our open-ended novelty framework within the CoSMoS approach by producing the following components (summarized in Figure 6):

1. A meta-model: given that our framework defines novelty with respect to a model and a meta-model, it is necessary to define a particular metamodel. The meta-model defined in this paper (Figure 3) is one possibility; others might be used to capture other forms of OE. Whatever meta-model is chosen, it must be sufficient to support the desired concepts of OE.
2. A domain model: an instantiation of the meta-model in domain terms, including desired emergent properties (where known), and excluding shortcuts.
3. A platform model: another instantiation of the meta-model related to the domain model in the following way:
   - The domain desired emergent properties are not present in the platform model, to ensure that they do not become explicitly coded into the simulator.
   - Once the necessary shortcuts have been determined, and have been argued not to preclude the desired emergence, the relevant domain components are replaced with their shortcut alternatives in the platform model
   - The remaining domain model components are transferred across to the platform model.

- Instrumentation and interfaces are added, to allow the simulator to be used to run open-ended novelty experiments; this might include emergent recognizers and model changers (see section 6.5), where appropriate.

4. A simulator built to the specification that is the platform model.
5. A results model: a further instantiation of the meta-model, suitable for capturing and analyzing observed emergent properties of the simulator in domain model terms.

The CoSMoS approach is a principled way to examine complex systems. There are other possible approaches that can be used to avoid confusion and to make sure that the novelties discovered in a simulation were not already built into the system from the outset. One should use some principled, coherent methodology, so that one can make meaningful claims about types of novelty.

7. Illustrative Examples

In this section we explain via examples from the literature how the model/metamodel approach captures the presence and absence of OE in both computational systems (Game of Life, section 7.1) and real world systems (experimental biological evolution, section 7.2), and in particular how it is all relative to a model/meta-model. Then we give examples of genetic algorithms (section 7.3), genetic programming (section7.4), and automata chemistries (section 7.5) to demonstrate some of the “shortcuts” described earlier, how they constrain the amount of novelty, and what consequences these shortcuts have for OE.

7.1. Game of Life

Conway’s Game of Life (GoL) (Berlekamp et al., 1982; Gardner, 1970) is a two-dimensional cellular automaton (CA), and a typical example quoted when talking of novelty and emergence. Here we show how GoL exhibits various kinds of novelty relative to a model instantiated from our meta-model. Other authors discuss emergence in GoL in different terms, for example (Gotts, 2009).

The base-level model has the following components: the cells, *atomic entities* at level 0 that can be in state ‘alive’ or ‘dead’; the entire GoL arena, an *aggregate system* comprising a collection of cells laid out in a grid pattern; the *environment*, considered as providing the space and time for this grid; and *interactions* between cells, which communicate their state to their eight nearest neighbors, and behave by synchronously updating their own state based on the GoL rules. At each timestep, if a cell is dead and has exactly three live neighbors, it becomes alive; if it is alive and has fewer than two or more than three live neighbors it dies.

As the CA executes, cells change state, exhibiting *variation*, that is, type-0 novelty (a change within the existing model only modifying the value of the existing state).

Rapidly, the observer notices the appearance of certain patterns: some localized groups of cells switch through cyclic patterns (on a longer time-scale). These localized groups are modeled as level-1 entities, comprising the relevant level-0 cells and the local grid environment over an extended number of update cycles. Thus the meta-model is augmented with a new level: level 1; and the model is augmented with a specific system entity type: an

oscillator (including period-1 oscillators called “still lives”). This modeling of oscillators is then an example of *emergence*, that is, type-2 novelty (a change to the meta-model adding a new level).

The observer also notices further patterns that seem to move across the grid, called “gliders” (or, more generally, “spaceships”). These are also modeled as level-1 entities, comprising the now-changing sets of cells and local grid environments that encapsulate these patterns. This modeling of gliders is then an example of *innovation*, that is, type-1 novelty (a change to the model adding a new entity type).

Then, the observer notices that these moving gliders *interact* by colliding: such interactions can produce new oscillators, gliders, and other patterns. This modeling of gliders is also an example of *innovation*, that is, type-1 novelty (a change to the model adding a new kind of entity, hence a new interaction type).

Eventually, the observer notices that these moving gliders and oscillators can be positioned to interact in such a way that they form a level-2 entity: a Turing Machine (Rendell, 2002). This model of Turing Machine in the GoL is then another example of *emergence*, that is, type-2 novelty (a change to the meta-model adding yet another new level).

In this example, the scientific model and meta-model are changed based on observations, hence innovation and emergence are observed. The possibility of innovation and emergence spring from the GoL engineering model that is itself not changed during the process (see figure 1).

7.2. Experimental evolution

Experimental evolution is a biological discipline where organisms evolve in controlled experiments, in order to explore natural evolution in real time. By defining experimental protocols, and preparing isolated strains of organisms for observation, shortcuts are introduced into the process, removing it from the natural evolutionary environment.

Since many predictions of evolutionary biology assume very large effective populations or many generations, most experimental evolution systems use viruses or bacteria, though other microorganisms (such as yeast and other fungi, nematodes, and more) and even some larger organisms (including fruit flies, mice, plants, and more) have also been used. The typical experiment involves several generations of the organism(s) under study, while selecting for some trait. Given a response to selection, which is nearly always present, the biologist can untangle the causes of the response using genetics, transcriptomics, metabolomics, or other techniques.

Experimental evolution is often “open-ended” only in that the experimenter does not specify or enforce pre-determined mechanisms for adaptation. However, this is not the kind of open-endedness we defined in this paper. Yet, innovation – in our use of the term – commonly emerges, such as when viruses adapt to a new host, new temperature, or other environmental change in an evolving viral/bacterial system. Some experimental evolution systems, such as Richard Lenski’s famous “long-term evolution experiment” have shown a clear open-ended dynamics (in our use of the term). Lenski has transferred over 62,000

generations of E. coli as of the time of this writing (personal communications), without (intentionally) selecting for any particular phenotype (Barrick et al., 2009). Different lineages of organisms with clearly distinct phenotypes and reproductive fitness have emerged during the course of this experiment. More recently, after over 27 years, a genuine speciation event may have arisen, where the bacteria have developed the ability to metabolize an entirely new energy source, which was always present in the media but never used: citrate (Barrick and Lenski, 2013; Blount et al., 2008). Other open-ended events have been observed in this experiment such as the observation of a stable polymorphism (Plucain et al., 2014) that may correspond to the emergence of a new ecological level. One of the main difficulties in experimental evolution is to understand how the "shortcuts" (e.g., serial transfer, culture medium, artificially stable environments...) are likely to trigger the observed OE events.

7.3. Genetic algorithms

The genetic algorithm (GA) (Goldberg, 1989; Holland, 1975) is another example. GAs have exhibited an astounding ability to produce creative and surprising solutions to technical optimization and design problems (Renner and Ekart, 2003).

GAs work by manipulating a set of parameters that code for the solution of an application problem, such as the minimization of a function. Each of these parameters might be considered a level-0 entity, with the set of parameters providing a solution being an entity at level 1. In GAs, this is called a *genome*.

At a still higher level in the GA (level 2) sits the so-called *fitness function*, a shortcut for helping to judge the quality of a solution. In biology, fitness is an implicit concept assigned retrospectively to individuals based on their reproductive success. In GAs, fitness is usually explicitly defined, while individuals are implicitly assumed to be produced by the genome. We described evolutionary computation above in terms of function minimization, where the difference between the function and what is required for the minimum (the *error*) is equivalent to (inverse) fitness and can be used to judge solution quality.

The list of fitness values can now be used to provide a selection signal from this level-2 system to level-1 entities by removing low-quality solutions and thus providing space for new individuals in the population generated by variation processes. The actual variation is produced by random[15] change on individual parameters (*mutation*) or by choosing sets of parameters from different individuals (*recombination*).

Often a set of discrete parameters can encode a solution for the problem and the number of dimensions does not change. Thus, the space of all possible combinations of parameters (and therefore solutions) is bounded. Yet, still interesting and novel solutions might emerge (despite using only type-0 novelties) as the combinatorial spaces under consideration quickly exceed the number of elementary particles in the universe. The limiting factor in GAs seems

[15] The variation is technically *pseudorandom*, being generated by a deterministic algorithm. This process is itself a lower level system in the model. Details of this level can significantly affect GA behavior.

to be the fixed fitness function, which leads the search sooner or later to stagnating fitness plateaus.

The situation with respect to shortcuts is depicted in Figure 7. A genetic algorithm as well as other similar evolutionary systems are highly constrained. Individuals (level 2) are organized in a population (level 3) with an entirely hardcoded structure that imposes selection on individuals. Individuals are made of data but some of their properties are hard-coded (individuality, replication). Individuals are made of genes that are chemical entities (level 1) producing the variation. Here, however, variation is hard-coded and cannot change. The generative rules (denoted by "A" in Figure 7) are also highly simplified: they are a fixed-size bitstring[16].

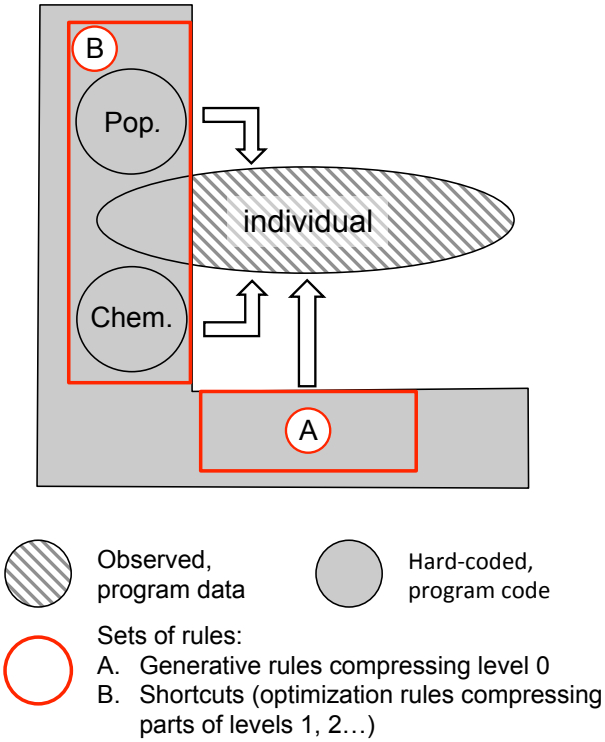

Figure 7: Simulation with shortcuts, GA example. In a GA, the level of interest is the individual. These individuals are grouped into a population and they are made of elements (genome, phenotype) that give them their dynamic. Now the population level is a mere aggregate of individuals (see section 6.3), hence not an explicit level in the hierarchy. Similarly, the elements that compose the individual and their relationships (e.g., the genotype-to-phenotype mapping) are explicitly encoded in the software. So the classical GA structure is a level 2 composed of individuals that evolve through variation events imposed by a fully shortcutted lower level and a selection pressure imposed by a fully shortcutted higher level. Note that for sake of clarity we distinguish the generative rules (A) from the lower level chemistry rules in box (B) but that the latter could have been merged with the former.

[16] Note that open-ended GAs could be implemented providing the bitstring length and the number of genes it encodes can evolve. In this case innovation events could be possible as observed, e.g. in (Knibbe et al., 2007).

7.4. Genetic programming

Genetic Programming (GP) (Banzhaf et al., 1998; Koza, 1992) is a computational method using similar principles as GAs. The individuals under variation and selection are computer programs or other active entities. Here, type-1 novelties make their regular appearance, as a result of the active nature of the required solution. Still, an explicit fitness function (often unchanging) governs the procedure and leads to improved solutions. Individuals in GP have to accommodate regular dimensional changes and should be better considered *behaviors* of their genomes (Foster, 2001).

Level-0 entities in this case are elementary behaviors that might be useful in the context of the problem. An example might be a GP system learning to play a game like chess. The combination of elementary behaviors makes up an individual able to play a game of chess. This would include the judgment of particular positional situations that would lead to different responses. These game-playing strategies would be level-1 entities subjected to type-0 and type-1 changes, again controlled by random mutations and recombinations.

Level 2 would be a more or less successfully played game. Here we can see that higher levels require the integration both in time and space of what is coded in the genome of an individual. Again, the constraints to the GP model are severe and tend to greatly limit the emergence of novelty.

In some forms of genetic programming, however, the mechanisms of variation are not fixed. For example, in “meta-genetic programming” a population of variation operators co-evolves with the main population (Edmonds, 1998; Kantschik et al., 1999), while in “autoconstructive evolution” systems the individuals in the main population are responsible both for problem solving and for producing offspring with variation (Spector, 2010; Spector and Robinson, 2002). In these systems the mechanisms of variation are themselves subject to variation and selection, and hence can evolve. The potential that these systems have for producing open-ended novelty is a subject for future study.

7.5. Coreworlds/Automata Chemistries

So-called “coreworlds” or “automata chemistries” denote a loosely defined set of related, but distinct, evolutionary computational systems. Examples include Coreworld (Rasmussen et al., 1990), Tierra (Ray, 1992), Avida (Adami and Brown, 1994) and Stringmol (Hickinbotham et al., 2016, 2010b). These are conceptually, or superficially, similar to GP systems, in that the Darwinian individuals are computational processes (executing programs) hosted in some shared, or at least interconnected, memory system[17]. However they should be contrasted with GP systems in a number of important respects.

First, replication is not implemented by shortcut: the processes are responsible for their own replication. Indeed, non-replicating or sterile processes can be instantiated by design, or can

[17] The “core” in “coreworld” derives from the earlier CoreWar programming game (Dewdney, 1987), and invokes the typical linear, random access, memory configuration originally associated with early magnetic core hardware.

arise through spontaneous perturbation, and can significantly influence evolutionary dynamics; a phenomenon which is not possible in most GP (or GA) systems where all individuals are directly replicated by shortcut, regardless of their intrinsic functionality. As one specific consequence of this self-replication, fitness is, to some signi_cant extent, intrinsic rather than extrinsic.

Second, the individual processes execute (pseudo-)concurrently, and their dynamic interaction typically constitutes another significant factor determining relative fitness.

Third, the programming formalism is usually based on a machine code representation (translated to an assembler-like representation in the user interface); as opposed to the high level formalisms (e.g., lisp) typically adopted in GP. This formulation makes self-modification/metaprogramming (the ability of programs to treat their own code, and that of other programs, as data to be operated on) very natural and easily accessible. In almost all empirical work with coreworlds, this metaprogramming forms the basis for implementing replication through a mechanism of direct self-inspection. However, this is not in any sense intrinsic to the underlying platforms, and some examples exist in the literature of implementing fully von-Neumann style self-reproduction with mutable GP-mapping instead (Baugh, 2015; Hasegawa, 2015; McMullin,

2012).

In terms of the framework presented here, all coreworlds have a base-level model (level-0) which includes one or more discrete random access memory system(s) and a dynamic (and indefinite) number of execution threads (CPUs). At this level, they generally support type-0 novelty through stochastic perturbations of memory contents. An individual process (level-1) is constituted by one or more CPUs coupled with some configuration (content) of one or more segments of memory. All coreworlds directly allow for type-1 novelty at level-1 (the spontaneous and open-ended introduction of new types of level-1 entities). This is typically triggered through the type-0 stochastic perturbation of memory contents at level-0; but can also be generated through (re-)writing the content of memory locations (i.e., by programmatic action). Either type of mechanism can lead to expansion (or contraction) of the memory segment(s) in use by any given process. That is, even if level-1 individuals are modelled as dynamic systems on a state space, the dimensionality of this state space can itself change.

Level-1 individuality in coreworlds (individually distinguished and identified executing processes) is normally shortcut to the extent that there is external “operating system” support to group together one or more execution threads with one or more memory segments, which the system monitors and logs as constituting individuals. However, the coherence of this shortcut may be blurred or undermined if the memory segments are drawn from a shared address space, because in that case individual processes may still potentially read, write, or execute memory segments that are *notionally* associated with a different process (or with none). This can have significant phenomenological consequences in particular coreworld systems.

Typical experiments in these systems are externally seeded with one or more such (level-1) individuals, which have been designed to have the property of self-replication. Further, the

self-replication mechanism is such that it supports the possibility of heritable mutation (type-1 novelty within the class of self-replicating level-1 individuals). Thus, the meta-model already allows for the development of populations (lineages) of level-1 individuals having essentially identical programmatic behavior (by virtue of identical initial memory content). These lineages are then level-2 entities that can be classified into distinct strains, which exhibit typical ecological and evolutionary phenomena. However, because these strains were already part of the meta-model, their appearance does not generally constitute type-2 novelty (emergence).

That said, an exception might be argued for the Amoeba coreworld system (Pargellis, 2001). In that case, it is not seeded with self-replication functionality at level-1, but that spontaneously emerges. The system was conceived and designed with that specific possibility in view, as have been several auto-constructive evolution systems (Spector, 2010; Spector and Robinson, 2002) Nonetheless it can be said that the meta-model initially lacks any coherent level-2 entities; but once the level-1 self-replication functionality emerges, then such population-lineage entities must be added to the meta-model (i.e., level-2 must be introduced) and therefore this is properly an example of type-2 novelty.

Returning to the case of coreworlds where self-replicating level-1 individuals are externally seeded, there are still some subtleties in classifying the types of novelty arising. In the case of the "original" Coreworld (Rasmussen et al., 1990), although seeded with coherent self-replicating level-1 individuals, the self-replicating functionality (and thus conventional Darwinian evolutionary dynamics at the level-2 of interacting lineages) is typically lost very quickly, due to unrestricted overwriting of memory segments notionally associated with different individuals[18]. This could be modeled as a type-2 novelty where the meta-model is changed by *losing* a level. By contrast, Tierra introduced a critical design feature of a "write protection" shortcut. This allows individuals to protect against external disruption of their own "allocated" memory segments, while still allowing them to pervasively read the contents of all memory (allocated or not). This ensures reliable persistence of the self-replication functionality and thus the level-2 of distinct strains. Unfortunately, in itself, this also allows a single self-replicating seed process to generate an exponentially growing lineage that quickly exhausts (allocates) the entire, finite, memory system. This would prevent any further replication (since no more memory segments can be allocated), and bring evolution (open ended or otherwise) to a halt. Accordingly, it was necessary to introduce a further, additional, shortcut, the so-called "reaper", which stochastically "kills" random processes, and deallocates their memory segments (and destroys their execution threads). This does allow Tierra to exhibit sustained type-1 *innovation* at the level-2 of strains, including phenomena of selection and diverse forms of parasitism.

In practice however, evolutionary novelty even in Tierra quickly "plateaus". While there is continuing generation of type-1 novelty at the level-1 of individuals, and this does continue to generate new, distinct, strains at level-2, these seem to exhaust the scope for any further distinctive ecological or selective behaviors at level-2. That is, there is no further even type-1

[18] Strictly, Coreworld does not incorporate a memory "allocation" shortcut per se at all.

novelty at level-2, and certainly no generation of identifiable, coherent, entities at any higher level (i.e., no instance of type-2 novelty of emergence).

The Avida system sought explicitly to open up further evolutionary potential in a coreworld system (as compared to Tierra) by adding the possibility for additional, externally specified, "behavioral challenges". These take the form of "tasks" that level-1 individuals may complete, and by doing so may enhance the Darwinian fitness of their corresponding lineages (strains). While this does lead to a diverse range of interesting phenomena, for the purposes of the current discussion, it does not seem to add any decisive new feature. In particular, the "plateauing" of novelty phenomenon is still observed. It is delayed somewhat while solutions are sought for the externally defined tasks. Once those tasks are satisfied, evolution again stagnates as, within the system itself, there is no mechanism for new tasks to arise. And, as with Tierra, there is no case of type-2 novelty (emergence of level-3 entities).

For the moment, it is an open (and active) research question whether the coreworld approach can be further developed in a way that would significantly extend its capability to exhibit open-ended evolution. As a minimum, any such development would seem to require mechanisms for generation of new selective "niches" (i.e. where selection, at the level-2 of strains, is not operating in just a single fitness landscape) and for the potential "individuation" of new entities at level-3.

8. Insights from the EvoEvo models

8.1. Introduction

During the EvoEvo project (especially WP3) several models have been used to investigate different aspects of the "Evolution of Evolution" process (variability, robustness and evolvability). Among these models one – Evo^{2}Sim – has been specifically designed and implemented for the project (see WP2 deliverables), hence using the approach proposed here (though the development of the model has been done in parallel with the theoretical developments, thus being less formalized). In the following section we will review the different aspects of Evo^{2}Sim that use the concepts proposed above as well as some of the results obtained with this model that shed light on the open-endedness concept. Then, in section 0, we will specifically discuss the emergence of spatio-temporal structures as a mechanism for open-ended evolution. Indeed, in various models used during the project we observed that the spatio-temporal interaction between evolving individuals is a fundamental factor in the emergence of new levels of organization in an evolving system. The generality of this claim has been tested on the StringMol model used in WP4 with very positive results (Hickinbotham and P Hogeweg, 2016).

8.2. Evo^{2}Sim

Description of the model:

Evo^{2}Sim (Rocabert et al., 2015, 2016a, b) is the multi-level model developed in the workpackage 2 of the EvoEvo project. It has been extensively described in deliverable D2.7

and released in deliverable D2.8. Our objective here is not to describe the model but to focus on its "open-ended" features and on the main results obtained in this context.

Evo^{2}Sim integrates many aspects of the architecture for open-ended simulations proposed in section 6, including several shortcuts (section 6.2). Figure 8 presents the general characteristics of the model. It is based on a generative layer (aka an artificial chemistry (Dittrich et al., 2001; Banzhaf and Yamamoto, 2015)) composed of several "molecular" elements all identified by ID-tags. These molecular elements can be:

- **Genomes:** In Evo^{2}Sim the genome is a circular strand with a coarse-grain encoding of coding and non-coding elements (pearl on a string formalism). A coding element can be functional or non-functional depending on its local neighborhood on the sequence (e.g. an "enzyme" pearl is functional iff is immediately follows a promoter, possibly flanked of binding sites).
- **Metabolites**: Metabolites are the primary resources and product of Evo^{2}Sim cells. In Evo^{2}Sim the metabolite universe is represented by natural integers. Metabolites can be transformed into one another by the enzymes. Importantly metabolites can be pumped in/out from the cells and diffuse in the environment (figure 8B.2). They thus implement an interaction canal between the individuals in the population. As discussed in section 6.2.3 this interaction canal is necessary to enable open-ended events to occur.
- **Enzymes:** Enzymes are encoded on the genome by mean of coarse-grain pearls activated by promoters (figure 8A.2 and 8A.3). They mediate all metabolic reactions in the cell through transformation of a metabolite into another (both being identified by their tags). Specific enzymes encode for in/out pumps that enable the cell to uptake/release metabolites from their environment. Thus enzymes and metabolites together form a metabolic network inside de cell (figure 8A.4) and all the metabolic networks of the populations are connected together through pump activity[19] and metabolites diffusion in the environment (figure 8B.2).
- **Transcription Factors:** As enzymes, transcription factors (TFs) are encoded on the genome. However, TFs don't code for the metabolic network. Rather they are able to bind on specific DNA pearls (the "Binding Sites") and to modulate the activity of the promoters that are flanked by these the binding sites. TFs thus form a regulation network that is able to modify the enzymatic composition of the cell, hence to adapt the metabolic network to the environment state. To interconnect the regulation network and the metabolic network, we allow TFs to be activated/inhibited by cofactors (i.e. specific metabolites that change the confirmation of the TFs when bound to them, thus activating/inhibiting them).

All these elements interact one with the others. Enzymes and Transcription Factors are decoded from the genome, enzymes and metabolites form the metabolic network, TFs, co-factors and enzymes form the regulation network and free metabolites compose the environment.

[19] Note that at cell death all the metabolic content of the cell is released in the environment. Thus cells can be indirectly connected even in the absence of outflow pumps.

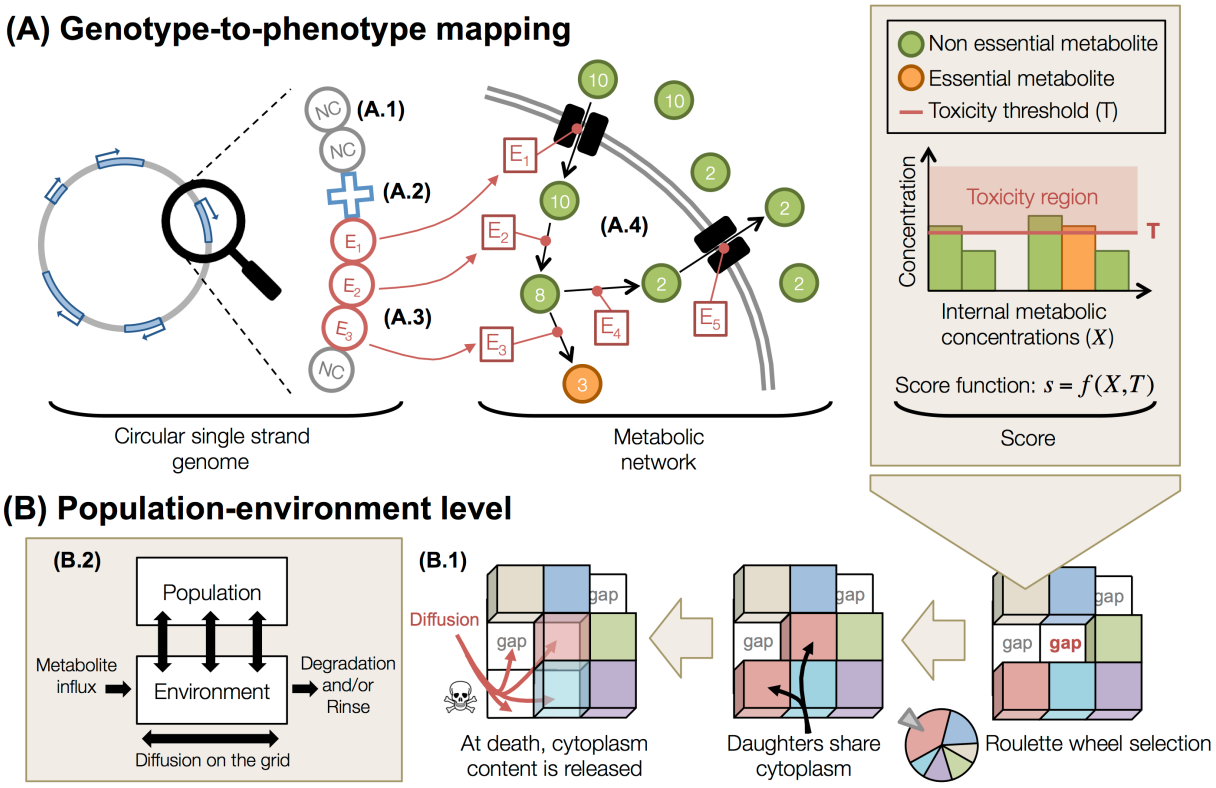

Figure 8. Overview of the Evo2Sim Model. For sake of simplicity the regulation network part is not displayed on the figure.

While the generative layer is composed of molecules, the central level of the model is the cellular level. The cell behavior is governed by the two molecular networks (metabolic network and regulation network) possibly interacting together and with the environment. In Evo^{2}Sim the cell level make an extensive use of shortcuts. In particular, the cell identity is shortcutted (section 6.2.1), so do the cell replication process (section 6.2.2) and the transcription-translation process: In Evo^{2}Sim cells are placed on a grid with a maximum of one cell per grid element (figure 8B.1). A cell can replicate only if one of its neighbors is empty. In this case the cell competes with all the living neighbors of the empty grid element and, if it wins the competition, it is allowed to divide and colonize the empty element. In accordance with what has been proposed in section 6.2.3 we designed an explicit fitness criterion to choose the individual that will be allowed to divide. This criterion is arbitrary: among all the metabolites produced by the cell, some are considered to favor cell replication[20] while others are “only” intermediate products. We will thus be able to test the claim that explicit fitness criteria do not prevent open-endedness providing the fitness of the individuals depends on the activity of the other individuals. This is indeed the case here since the activity of the metabolic network (hence the production of essential metabolites) is directly dependent on the enzymatic composition of the environment that is itself dependent on the activity of the other cells, on the metabolic influx and on the diffusion process (figure 8B.2).

[20] In the current version of Evo^{2}Sim, these “essential metabolites” are those that have a prime number ID-tag.

Results:

In terms of our metamodel (section 3.2), Evo^{2}Sim is composed of two levels: the molecular level (genomes, metabolites, enzymes and transcription factors) and the cellular level (metabolic and regulation networks). There is no higher level since the population is composed of undifferentiated individuals that all play the same role. Thus the population constitute an *aggregate system* but not and *entity* (see section 3.2.4). Indeed, evolutionary theory predicts that in such a situation, the niche exclusion principle will be fully active and the population will ultimately be composed of a unique quasispecie originating from a single master sequence.

As discussed in section 7.2 experimental evolution setups may show open-ended evolution. In particular, in Richard Lenski's LTEE experiment (Barrick et al., 2009) several open-ended events have been observed. In particular, the diversification event has been observed in one of the lineage (Plucain et al., 2014), in apparent contradiction with the niche exclusion principle. To test whether Evo^{2}Sim is able to show the same kind of events, we used it to replicate the experimental setup of the LTEE: populations in Evo^{2}Sim were submitted to seasonal environments (i.e. regular succession of feasts and famines). Controls were composed of similar populations submitted to constant environments (~chemostats) and to random succession of feasts and famines. Results (Rocabert et al., 2016a, b) show that the seasonal environment strongly favors the emergence of stable polymorphism (figure 9) and, importantly, that this polymorphism is due to the co-existence of two different ecotypes in the population. These ecotypes survive by exchanging metabolites, one specie consuming the byproducts of the other (niche construction).

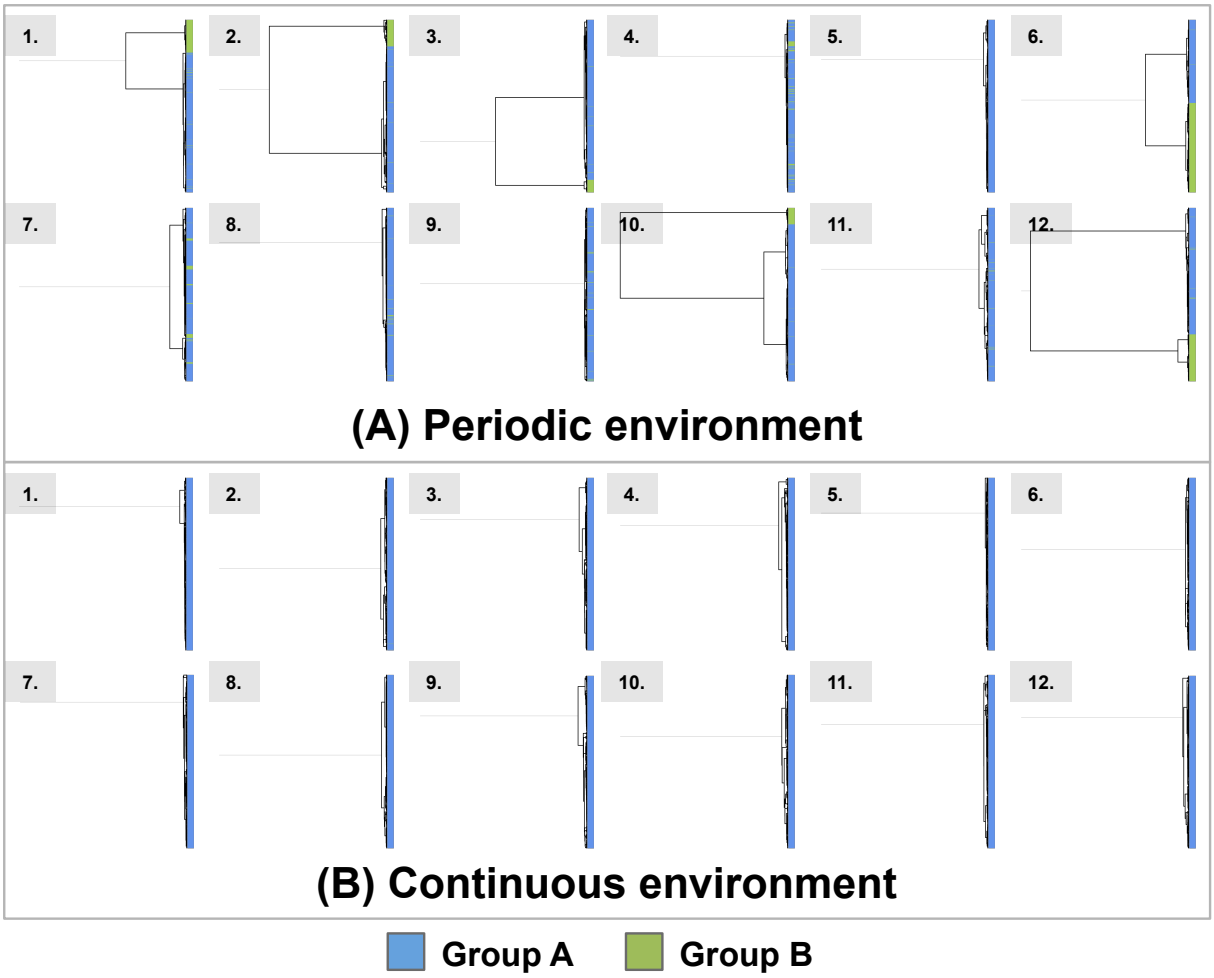

Figure 9: phylogenetic trees of the populations. (A) Populations evolved in the seasonal environment mimicking the LTEE experiment. (B) Populations evolved in the constant environment mimicking a chemostat. In situation A, half of the populations (pop 1, 2, 3, 6, 10 and 12) show long-term divergence between two phylogenetic groups. This situation is never observed in environment B. Moreover, in all the cases where long-term divergence is observed, the two subtrees correspond to different ecotypes. Here group A (blue) correspond to individuals consuming mainly the primary resource provided by the experimental setup (i.e. the glucose in the case of the LTEE). Group B (green) correspond to individuals that are not able to grow on the primary resource but that consume the byproducts released by individuals of group A.

In terms of open-endedness, this stable polymorphism corresponds to the addition of a new organization level to the meta-model, hence to a type-2 novelty, an *emergence*. This new level is the ecological level. It is composed of interacting species interacting through a “food web”. Note that this emergence does not correspond to a “major transition” (Maynard-Smith and Szathmary, 1995) since the new entities (the “species” of the “ecosystem”) are not able to reproduce on their own. Interestingly, this result enables us to identify a new kind of shortcut that has not been envisioned during of theoretical thinking. Indeed, the temporal structure of the system is here partly imposed by the practitioner (ourselves in the context of Evo^{2}Sim, Richard Lenski and collaborators in the context of the LTEE). It appears that some forms of temporality are more likely to lead to the emergence of stable ecosystems than

others. Hence, temporal forcing of the environment could be considered as a shortcut. This shed a new light on the LTEE and on experimental evolution in general by showing that how practitioners can involuntarily influence the outcome of the experiment when setting the experimental design.

8.3. Emergence of spatio-temporal structures

Two hallmarks of open-ended evolution are the (1) emergence of novel higher-level "Darwinian" entities and (2) the emergence of new directions of selection, and therewith novel structures at the basic lower level. We have achieved both these hallmarks in spatial systems in which fitness is not explicitly defined, but reflects replication/death rates, of interacting entities. Such interaction can be predator-prey, host-virus interactions etc. We have done most our experiments in RNA-like systems in which a replicase catalyses the replication of other entities. This involves complex formation, and copying. These processes can be defined in a variety of ways, ranging from very simple models in which binding and copying are only defined as parameters (Takeuchi and Hogeweg, 2009; Colizzi and Hogeweg, 2016a,b), to models with structured entities (e.g. RNA strings (Takeuchi and Hogeweg, 2008; , and Hogeweg, 2014), or programs (Hickinbotham and Hogeweg, 2016)) in which binding depends on complementary matching, and replicase activity depends on the structure of the entity (RNA folding, or program execution), In the former case, mutations simply change the binding parameters, in the latter case the structure of the entities, and therewith the genotype to phenotype mapping. Independent of the implementation, all these systems have the following properties in common.

In well-mixed systems, which for the simple models can be defined as ordinary differential equations (ODE's) such replicase systems evolve to extinction. This is the case because giving catalysis (being a replicase) is an "altruistic" property, costing time and leading to a benefit for the other entity and not for 'self'. Thus, replication activity is minimized and the system goes extinct. This evolution towards extinction is prevented in spatial systems, through spatial self-organization into traveling waves. Like in the well-mixed system replicase activity is minimized and/or parasite activity is maximized, where parasites are entities that have lost catalytic activity, but can be replicated, and are indeed optimized for being replicated. The waves consist of replicases in the front, invading empty space and parasites in the back being replicated more efficiently by the replicases and invading the area of the replicases, and leaving empty space behind. These waves are higher order Darwinian entities "with a life of their own". They replicate, mature, die, and compete with other waves, and they maximize their 'fitness' by either maximizing their birthrate or minimizing their death rate. Either way, they need sufficient catalytic activity of the entities of which they are composed to survive. Thus, once these waves form two opposing selection pressures operate: the one minimizing catalysis for the individuals, and the one increasing catalysis for the waves. Such emerging multilevel evolutionary systems mediate an escape from evolutionary self-destruction, which is of course a prerequisite for further (open-ended) evolution. Indeed, such multilevel 'survival' opens up new dimensions for selection for the lower level entities as well.

A striking example of this 'multilevel open-endedness' was realized by implementing the StringMol system (Hickinbotham et al., 2010a,b) in space (Hickinbotham and Hogeweg

2016). A replicase in StringMol is a program, coded as a string, which executes the replication (copying) of strings to which it binds, one opcode at the time. This representation implements evoevo directly because novel mutational operators evolve through the evolution of the replicators, which introduces novel “errors” in the copying process. In earlier experiments with StringMol the evolutionary potential of the system was severely hampered by the emergence and take-over of parasites, i.e. short strings which were replicated very fast, but could not copy others, thus leading to rapid extinction of the system. Embedding the system in space, the parasites also arise, but do not lead to extinction, by the processes explained above: traveling waves arise, which invade empty space, whereas short very fast replicating parasites invade and extinguish the waves, but the system as a whole survives and the replicases can continue to evolve.

This continued evolution generates very “clever” and surprising strategies of the replicases to “defend” themselves against the parasites. This renders survival less dependent on the wave structures (which cause and require local extinction) and leads to increased population size, as is seen in figure 10, where the spatial structure at an earlier and later time are shown. Whereas the primary selection pressure on replicases is to become shorter in order to replicate faster, we see very long replicases at intermediate times. In the end replicases become shorter again, but increase replication time, by scanning themselves before they start replicating. This turns out to be an efficient anti-parasite strategy, as it reduces the advantage of short parasites, because replication time is always strongly dependent on the length of the replicase, whatever the length of the template/parasite is. Thus, by means of, and on top of the emergence of novel higher order entities, which their own survival strategies, preventing the collapse of the system, the lower level entities are freed to evolve in novel directions.

Another example of such a scenario of emergence of multilevel evolution, and the opening up of new dimensions for evolution is the evolution of DNA in the RNA world. DNA lengthens the replication cycle (and is therefore disadvantageous, but through division of labour between information storage and information usage prevents evolutionary deterioration (Takeuchi et al., 2011). The division of labour between information storage and usage (DNA and RNA) is indeed a pre-eminent example a major transition in biological evolution, and can thus be explained as mediated by multilevel evolution. In spatially embedded systems novel multilevel evolution emerges through self-organization. However, also when higher levels of selection are predefined (e.g. as protocells with embedded replicase systems), opposing levels of selection can mediate survival, as well as the emergence of novel evolutionary direction. (Takeuchi and Hogeweg, 2009; 2016).

We conclude that embedding interacting replicating systems in space, automatically generates multilevel evolutionary systems, which in turn opens up novel directions in evolution generating stepping stones to open ended evolution.

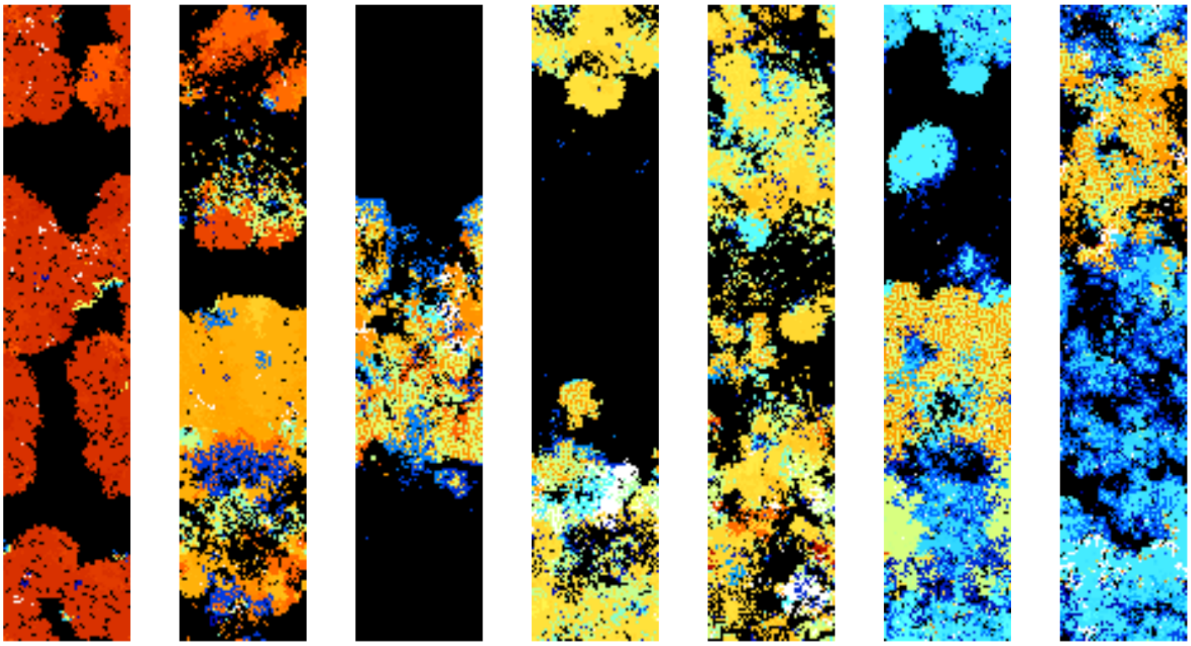

Figure 10: Spatial configuration at t=10000, t=100000, t=300000, t=500000, t=700000, t=900000 and t=1000000. The strains are coloured by length, from 0 (blue) to 70 (dark red) and > 70 (white).

9. Conclusions

9.1. Summary

In brief summary, our argument is:

- Models and meta-models are used for building both scientific and engineering models of systems (they may be implicit)
- We define types of novelty and open-endedness *with respect to the system's current model and meta-model*:
    - Novelty in an observed system is classified as:
        1. Variation: novelty within the model
        2. Innovation: novelty that changes the model
        3. Emergence: novelty that changes the meta-model
    - *Open-ended event*: an event that results in innovation or emergence
    - *Open-ended system*: a system with the ability *to continually produce open-ended events*
- We provide one particular meta-model, incorporating the concept of levels, which allows it to capture "major transitions" as emergent events
- We introduce simulation "shortcuts": hard-coded behaviors at different levels needed to implement effective computer simulations
- We describe some illustrative existing systems, either from the literature (section 7) or from the EvoEvo project (section 8) in terms of their types of novelties and shortcuts

9.2. Discussion

The constant emergence of novelty from complex systems is a fundamental phenomenon of nature, societies, and computer simulations for which one might desire a unified theory. However, as we have shown, it is difficult to precisely characterize a notion of open-endedness that works in all cases, so that a general theory of open-endedness is hard to come by. The barriers are both conceptual and computational, as we have highlighted in the boxes above. However, we have argued that it is possible, and even useful, to attempt such a general theoretical framework, without committing *a priori* to any particular application domain.

One function of a general theory would be to develop domain independent models of open-endedness, so that predictions and experiments in one domain might support cross-domain conclusions. We have presented a meta-model for open-endedness that we have argued can encompass similarities between models of open-ended evolution in biological systems, in computer simulations and in technical systems as we intend in the EvoEvo project. We have applied this meta-model specifically to the unbounded emergence of novelty in computer simulations, with sufficient software engineering detail to show that our theoretical aspirations do not sacrifice practical applications.

A general theory should also provide a dictionary that would help us translate results from one domain into the other. Computational models based on a general theory of open-endedness should share enough with biological models to lay out exactly how computer simulations of open-endedness inform hypotheses about the emergence of novelty in biology. There is an established field of artificial society simulation, and simulations of biological phenomena are legion. Few of these computer simulations, however, explicitly present a model for the emergence of novelty together with a theoretical argument that the model is the same for both the simulation and the system being simulated.

We have argued that such a general theory of open-endedness is possible, and that it can lead to meta-models of the emergence of novelty that are appropriate for computer simulations. We do not claim, however, that the specific metamodel we present is always the most appropriate for a given analysis. Nor have we demonstrated how our computer simulation model is appropriate beyond the domain of computer simulations. There is probably no single meta-model for all questions, even if one has a general theory of open-endedness. Nevertheless, our hope is that attempting to develop such a theory, while remaining cognizant of the specific models that it would support, can be useful for prediction and experimentation across multiple domains.

10. References

C. Adami and C. T. Brown (1994) Evolutionary learning in the 2D artificial life system avida. *In Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 377-381

P. Amar, G. Legent, M. Thellier, C. Ripoll, G. Bernot, T. Nystrom, M. Saier, and V. Norris. (2008) A stochastic automaton shows how enzyme assemblies may contribute to metabolic efficiency. *BMC Systems Biology*, 2:27

P. W. Anderson (1972) More is different. *Science*, 177(4047):393-396

P. S. Andrews, F. A. C. Polack, A. T. Sampson, S. Stepney, and J. Timmis (2010) *The CoSMoS process, version 0.1: A process for the modelling and simulation of complex systems.* Technical Report YCS-2010-453, Department of Computer Science, University of York, Mar. 2010.

P. S. Andrews, S. Stepney, T. Hoverd, F. A. Polack, A. T. Sampson, and J. Timmis (2011) CoSMoS process, models, and metamodels. *In CoSMoS 2011: Proceedings of the 2011 Workshop on Complex Systems Modelling and Simulation*, pp. 1-14

P. S. Andrews, S. Stepney, and J. Timmis (2012) Simulation as a scientific instrument. *In Proceedings of the 2012 Workshop on Complex Systems Modelling and Simulation, Orleans, France*, pp. 1-10.

W. Banzhaf, P. Nordin, R. Keller, and F. Francone (1998) *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.

W. Banzhaf and L. Yamamoto (2015) *Artificial Chemistries*. MIT Press, 2015.

W. Banzhaf, G. Beslon, R. Doursat and S. Stepney (2016) Open-Endedness: Definitions and Shortcuts. *Second Open-Ended Evolution Workshop, Cancùn (Mexico),* July 2016, 2 p.

W. Banzhaf, B. Baumgaertner, G. Beslon, R. Doursat, J. A. Foster, B. McMullin, V. Veloso de Melo, T. Miconi, L. Spector, S. Stepney and R. White (2016) Defining and Simulating Open-Ended Novelty: Requirements, Guidelines, and Challenges. *Theory in Biosciences*, 135(3):131-161.

T. Baptista and E. Costa (2013) Step evolution: Improving the performance of open-ended evolution simulations. *In IEEE Symposium on Artificial Life, Singapore 2013*, pp. 52-59

N. A. Barricelli (1962) Numerical testing of evolution theories: Part i theoretical introduction and basic tests. *Acta Biotheoretica*, 16(1-2):69-98

J. E. Barrick and R. E. Lenski (2013) Genome dynamics during experimental evolution. *Nature Reviews Genetics*, 14:827-839

J. E. Barrick, D. S. Yu, S. H. Yoon, H. Jeong, T. K. Oh, D. Schneider, R. E. Lenski, and J. F. Kim (2009) Genome evolution and adaptation in a longterm experiment with Escherichia coli. *Nature*, 461:1243-1247

B. Batut, C. Knibbe, G. Marais, and V. Daubin (2014) Reductive genome evolution at both ends of bacterial population size spectrum. *Nature Reviews Microbiology*, 12(12):841-850

D. Baugh (2015) Implementing von Neumann's architecture for machine self reproduction within the tierra artificial life platform to investigate evolvable genotype-phenotype mappings. *PhD, Dublin City University. School of Electronic Engineering*, Nov. 2015.

M. A. Bedau (1991) Can biological teleology be naturalized? *The Journal of Philosophy*, 88:647-655

M. A. Bedau (1996) The nature of life. *In: M. Boden, editor, The Philosophy of Artificial Life*, Oxford University Press, 1996, pp. 332-357

M. A. Bedau (1999) Can unrealistic computer models illuminate theoretical biology? *In Proceedings of the 1999 Genetic and Evolutionary Computation Conference,* pp. 20-23.

M. A. Bedau, J. S. McCaskill, N. H. Packard, S. Rasmussen, C. Adami, D. G. Green, T. Ikegami, K. Kaneko, and T. S. Ray (2000) Open problems in artificial life. *Artificial Life*, 6(4):363-376

M. A. Bedau and N. H. Packard (1992) Measurement of evolutionary activity, teleology, and life. *In C. Langton, C. Taylor, D. Farme, and S. Rasmussen, editors, Artificial Life II*, Addison-Wesley, 1992, pp. 431-461

M. A. Bedau, E. Snyder, and N. H. Packard (1998) A classification of long-term evolutionary dynamics. *In proceedings of ALife IV*, pp. 228-237

P. J. Bentley (2003) Evolving fractal gene regulatory networks for robot control. *In ECAL 2003, volume 2801 of LNCS*, pp. 753-762

E. R. Berlekamp, J. H. Conway, and R. K. Guy (1982) *Winning Ways for Your Mathematical Plays, Volume 2: games in particular*. Academic Press, 1982.

R. Bianco and S. Nolfi (2004) Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce. *Connection Science*, 16(4):227-248

Z. D. Blount, C. Z. Borland, and R. E. Lenski (2008) Historical contingency and the evolution of a key innovation in an experimental population of Escherichia coli. *PNAS*, 105:7899-7906

L. Buss (1987) *The Evolution of Individuality*. Princeton University Press, 1987

A. Channon (2001) Passing the ALife test: Activity statistics classify evolution in Geb as unbounded. *In ECAL'01: Proceedings of the Sixth European Conference on Artificial Life*, pp. 417-426

A. Channon (2003) Improving and still passing the ALife test: Component-normalised activity statistics classify evolution in Geb as unbounded. *In Proceedings of Artificial Life VIII*, pp. 173-181

A. D. Channon and R. I. Damper (2000) Towards the evolutionary emergence of increasingly complex advantageous behaviors. *International Journal of Systems Science*, 31(7):843-860

E. S. Colizzi and P. Hogeweg (2014) Evolution of functional diversification within quasispecies. *Genome biology and evolution*, 6(8):1990-2007

E. S. Colizzi and P. Hogeweg (2016a) Parasites Sustain and Enhance RNA-Like Replicators through Spatial Self-Organisation. *PLoS Computational Biology*, 12(4):e1004902.

E. S. Colizzi and P. Hogeweg (2016b) High cost enhances cooperation through the interplay between evolution and self-organisation. *BMC Evolutionary Biology*, 16:31

A. K. Dewdney (1987) Computer recreations: A program called mice nibbles its way to victory at the first core wars tournament. *Scientific American*, 256(1):8-11

P. Dittrich, J. Ziegler, and W. Banzhaf (2001) Artificial chemistries: A review. *Artificial Life*, 7:225-275

A. Droop and S. Hickinbotham (2012) A quantitative measure of non-neutral evolutionary activity for systems that exhibit intrinsic fitness. *Artificial Life XIII*, pp. 45-52

B. Edmonds (1998) Meta-genetic programming: Co-evolving the operators of variation. *CPM Report 98-32, Centre for Policy Modelling, Manchester Metropolitan University (UK)*, 1998.

J. D. Fernandez, D. Lobo, G. M. Martn, R. Doursat, and F. J. Vico (2012) Emergent diversity in an open-ended evolving virtual community. *Artificial Life*, 18(2):199-222

C. Fernando, G. Kampis, and E. Szathmary (2011) Evolvability of natural and artificial systems. *Procedia Computer Science*, 7:73-76

J. A. Foster (2001) Computational genetics: Evolutionary computation. *Nature Reviews Genetics*, 2:428-436

R. Frigg and S. Hartmann (2012) Models in science. *In E. N. Zalta, editor, The Stanford Encyclopedia of Philosophy*. Edition, 2012.

M. Gardner (1970) Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223(4):120-123,

D. E. Goldberg (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989

N. M. Gotts (2009) Ramifying feedback networks, cross-scale interactions, and emergent quasi individuals in Conway's Game of Life. *Artificial Life*, 15(3):351-375

I. Harvey (1992) Species adaptation genetic algorithms: A basis for a continuing SAGA. *In Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 346-354

T. Hasegawa (2015) *On the evolution of genotype-phenotype mapping: exploring viability in the Avida artificial life system*. PhD, Dublin City University. School of Electronic Engineering, Mar. 2015.

F. Heylighen (2012) Brain in a vat cannot break out. *Journal of Consciousness Studies*, 19(1-2):126-142

S. J. Hickinbotham, E. Clark, S. Stepney, T. Clarke, A. Nellis, M. Pay, and P. Young (2010a) Diversify from a monoculture : effects of mutation-on-copy in a string-based artificial chemistry. In: Proc. of ALife Alife XII Conference, Odense, Denmark, pp. 24-31

S. Hickinbotham, E. Clark, S. Stepney, T. Clarke, A. Nellis, M. Pay, and P. Young (2010b) *Specification of the stringmol chemical programming language version 0.2*. Technical Report YCS-2010-458, Univ. of York, 2010.

S. Hickinbotham, E. Clark, A. Nellis, S. Stepney, T. Clarke, and P. Young (2016) Maximising the adjacent possible in automata chemistries. *Artificial Life*, 22(1)

S. Hickinbotham, P. Hogeweg (2016) Evolution towards extinction in replicase models: inevitable unless... *2nd EvoEvo Workshop, Amsterdam (NL), September 2016*, 5 p.

S. Hickinbotham, P. Hogeweg and S. Stepney (2016) Open-endedness in EvoEvo models (in prep)

J. H. Holland (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

C. Horsman, S. Stepney, R. C. Wagner, and V. Kendon (2014) When does a physical system compute? *Proceedings of the Royal Society A*, 470(2169):2014.0182

T. Hoverd and S. Stepney (2011) Energy as a driver of diversity in open-ended evolution. *In ECAL 2011, Paris, France, August 2011*, pp. 356-363

P. Huneman (2012) Determinism, predictability and open-ended evolution: lessons from computational emergence. *Synthese*, 185(2):195-214

T. J. Hutton (2002) Evolvable self-replicating molecules in an artificial chemistry. *Artificial Life*, 8(4):341-356

K. Kaneko (1994) Chaos as a source of complexity and diversity in evolution. *Artificial Life*, 1(1 2):163-178

W. Kantschik, P. Dittrich, M. Brameier, and W. Banzhaf (1999) Meta-evolution in graph GP. *In R. Poli, P. Nordin, W. Langdon, and T. Fogarty, editors, Genetic Programming: Proceedings EuroGP 1999*, pp. 15-28

M. Kimura (1984) *The neutral theory of molecular evolution*. Cambridge University Press, 1984.

A. Kleppe, J. Warmer, and W. Bast (2003) *MDA Explained: the Model Driven Architecture: practice and promise*. Addison-Wesley, 2003.

C. Knibbe, A. Coulon, O. Mazet, J.-M. Fayard, and G. Beslon (2007) A longterm evolutionary pressure on the amount of noncoding DNA. *Molecular Biology and Evolution*, 24(10):2344-2353

A. Koestler (1970) Beyond atomism and holism: the concept of the holon. *Perspectives in Biology and Medicine*, 13(2):131-154

J. Koza. *Genetic Programming*. MIT Press, 1992.

D. Lan (2006) Hierarchy, complexity, society. *In D. Pumain, editor, Hierarchy in Natural and Social Sciences*, pp. 81-119. Springer, 2006.

J. Lehman and K. O. Stanley (2011) Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189-223

J. Lehman and K. O. Stanley (2012) Beyond open-endedness: Quantifying impressiveness. *ALIFE XIII*, pp. 75-82

C. Maley (1999) Four steps toward open-ended evolution. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, volume 2, page 1336

O. Markovitch, D. Sorek, L. T. Lui, D. Lancet, and N. Krasnogor (2012) Is there an optimal level of open-endedness in prebiotic evolution? *Origins of Life and Evolution of Biospheres*, 42(5):469-474

J. Maynard-Smith (1988) Evolutionary progress and levels of selection. *In M. Nitecki, editor, Evolutionary Progress, pp. 219-230*. Univ. of Chicago Press, 1988.

J. Maynard-Smith and E. Szathmary (1995) *The Major Transitions in Evolution*. Oxford University Press, 1995.

J. P. McCutcheon and N. A. Moran (2012) Extreme genome reduction in symbiotic bacteria. *Nature Reviews Microbiology*, 10(1):13-26

B. McMullin (2012) Architectures for Self-reproduction: Abstractions, Realisations and a Research Program. *In C. Adami, D. M. Bryson, C. Ofria, and R. Pennock, T., editors, Artificial Life XIII*, pp. 83-90

D. Medernach, T. Kowaliw, C. Ryan, and R. Doursat (2013) Long-term evolutionary dynamics in heterogeneous cellular automata. *In Proceedings of the 15th annual conference on Genetic and evolutionary computation (GECCO'13)*, pp. 231-238

C. L. Nehaniv, J. Hewitt, B. Christianson, and P. Wernick (2006) What software evolution and biological evolution don't have in common. *In Second International IEEE Workshop on Software Evolvability (SE'06)*, pp. 58-65

F. Odling-Smee, K. Laland, and M. Feldman (2003) *Niche Construction: The Neglected Process in Evolution*. Princeton University Press, 2003.

A. N. Pargellis (2001) Digital Life Behavior in the Amoeba World. *Artificial Life*, 7(1):63-75

J. Plucain, T. Hindré, M. Le Gac, O. Tenaillon, S. Cruveiller, C. Médigue, N. Leiby, W. R. Harcombe, C. J. Marx, R. E. Lenski, and D. Schneider (2014) Epistasis and allele specificity in the emergence of a stable polymorphism in *escherichia coli*. *Science*, 343:1366-1369

K. Popper (1982) *The Open Universe: An Argument for indeterminism*. Hutchinson, 1982.

S. Rasmussen, L. Chen, D. Deamer, D. C. Krakauer, N. H. Packard, P. F. Stadler, and M. A. Bedau (2004) Transitions from nonliving to living matter. *Science*, 303(5660):963-965

S. Rasmussen, C. Knudsen, R. Feldberg, and M. Hindsholm (1990) The coreworld: Emergence and evolution of cooperative structures in a computational chemistry. *Physica*, 42D:111-134

T. S. Ray (1992) An approach to the synthesis of life. *In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, Artifical Life II*, pp. 371-408

T. S. Ray (1992) *Evolution, ecology and optimization of digital organisms*. Working paper 92-08-042, Santa Fe Institute, 1992.

P. Rendell (2002) Turing universality of the Game of Life. *In A. Adamatzky, editor, Collision-Based Computing.* Springer, 2002.

G. Renner and A. Ekart (2003) Genetic algorithms in computer aided design. *Computer-Aided Design*, 35(8):709-726

B. Rensch (1959) *Evolution above the species level*. Methuen, London, 1959.

C. W. Reynolds (1987) Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25-34

C. Rocabert, C. Knibbe and G. Beslon (2015) Towards an Integrated Evolutionary Model to Study Evolution of Evolution. *First EvoEvo Workshop, York (UK), July 2015, 15 p.*

C. Rocabert, C. Knibbe, J. Consuegra, D. Schneider and G. Beslon (2016a) Beware Batch Culture: Seasonality and Niche Construction Predicted to Favor Bacterial Adaptive Diversification. *Under review.*

C. Rocabert, C. Knibbe, J. Consuegra, D. Schneider and G. Beslon (2016b) In-silico experimental evolution highlights the influence of environmental seasonality on bacterial diversification. *2nd EvoEvo Workshop, Amsterdam (NL), September 2016*, 3 p.

R. Rosen (1991) *Life itself: a comprehensive inquiry into the nature, origin, and fabrication of life*. Columbia University Press, 1991.

K. Ruiz-Mirazo and A. Moreno (2012) Autonomy in evolution: from minimal to complex life. *Synthese*, 185:21

K. Ruiz-Mirazo, J. Pereto, and A. Moreno (2004) A universal definition of life: Autonomy and open-ended evolution. *Origins of Life and Evolution of Biospheres*, 34(3):323-346

K. Ruiz-Mirazo, J. Umerez, and A. Moreno (2008) Enabling conditions for 'openended evolution'. *Biology & Philosophy*, 23(1):67-85

E. Schrödinger (1944) *What is Life? The Physical Aspect of the Living Cell*. Cambridge University Press, 1944.

R. Schulman, B. Yurke, and E. Winfree (2012) Robust self-replication of combinatorial information via crystal growth and scission. *PNAS*, 109(17):6405-6410

M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauer (1997) A phylogenetic, ontogenetic, and epigenetic view of bioinspired hardware systems. *IEEE Trans. Evolutionary Computation*, 1(1):83-97

M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauer (1998) An introduction to bioinspired machines. *In Bio-inspired computing machines towards novel computational architectures, pp. 1-12. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998*

A. Skusa and M. A. Bedau (2002) Towards a comparison of evolutionary creativity in biological and cultural evolution. *In ALife VIII*, page 233-242

L. B. Soros and K. O. Stanley (2014) Identifying necessary conditions for openended evolution through the artificial life world of Chromaria. *Artificial Life 14: International Conference on the Synthesis and Simulation of Living Systems*, pp. 793-800

L. Spector (2010) Towards practical autoconstructive evolution: Self-evolution of problem-solving genetic programming systems. *In R. Riolo, T. Mc-Conaghy, and E. Vladislavleva, editors, Genetic Programming Theory and Practice VIII, volume 8 of Genetic and Evolutionary Computation*, pp. 17-33

L. Spector and A. Robinson (2002) Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3:7-40

R. K. Standish (2003) Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(2):167-175

S. Stepney (2012) A pattern language for scientific simulations. *In Proceedings of the 2012 Workshop on Complex Systems Modelling and Simulation*, Orleans, France, pp. 77-103

S. Stepney, K. Alden, J. L. B. Paul S. Andrews, A. Droop, T. Ghetiu, T. Hoverd, F. A. C. Polack, M. Read, C. G. Ritson, A. T. Sampson, J. Timmis, P. H. Welch, and A. F. T. Winfield (2016) Engineering Simulations as Scientific Instruments. (*in prep*).

S. Stepney and P. S. Andrews (2015) CoSMoS special issue editorial. *Natural Computing*, 14:1-6.

S. Stepney and T. Hoverd (2011) Reflecting on open-ended evolution. *In ECAL '11: Proceedings of the Eleventh European Conference on Artificial Life*, pp. 781-788

S. Stepney and G. Beslon (2016) Open-Ended Evolution: Definitions and Shortcuts. *2nd EvoEvo Workshop, Amsterdam (NL), September 2016*, 5 p.

E. Szathmary (2015) Toward major evolutionary transitions theory 2.0. *PNAS*, 112(33):10104-10111

N Takeuchi and P Hogeweg (2008) Evolution of complexity in RNA-like replicator systems. *Biology Direct*, 3(11)

N. Takeuchi and P. Hogeweg (2009) Multilevel selection in models of prebiotic evolution II: a direct comparison of compartmentalization and spatial self-organization. *PLoS Computational Biology*, 5(10):e1000542

N. Takeuchi, P. Hogeweg and E.V. Koonin (2011) On the origin of DNA genomes: evolution of the division of labor between template and catalyst in model replicator systems. . *PLoS Computational Biology*, 7(3):e1002024

N. Takeuchi, K. Kaneko and P. Hogeweg (2016) Evolutionarily stable disequilibrium: endless dynamics of evolution in a stationary population. *Proc. R. Soc. B*, 283(1830):20153109

T. Taylor (1999) *From artificial evolution to artificial life*. PhD thesis, The University of Edinburgh, July 1999.

T. Taylor, M. Bedau, A. Channon, D. Ackley, W. Banzhaf, G. Beslon, E. Dolson, T. Froese, S. Hickinbotham, T. Ikegami, B. McMullin, N. Packard, S. Rasmussen, N. Virgo, E. Agmon, E. Clark, S. McGregor, C. Ofria, G. Ropella, L. Spector, K. O Stanley, A. Stanton, C. Timperley, A. Vostinar, M. Wiser (2016) Open-Ended Evolution: Perspectives from the OEE1 Workshop in York. *Artificial Life* 22:408-423

C. Waddington (2008) Paradigm for an evolutionary process. *Biological Theory*, 3:258-266

M. Weisberg (2013) *Simulation and Similarity: Using Models to Understand the World*. Oxford University Press, 2013.

D. Wilson (1997) Biological communities as functionally organized units. *Ecology*, 78:2018-2024

W. Wimsatt (1987) False models as means to truer theories. *In N. Nitecki and A. Hoffman, editors, Neutral Models in Biology*, page 23-55. Oxford University Press, 1987.

E. Winsberg (2010) *Science in the Age of Computer Simulation*. Chicago University Press, 2010.